Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Machine Learning en la industria del automóvil

Autor: Pablo Lázaro Enguita

Tutor/es: Francisco Javier Rodríguez Sánchez

2018

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Machine Learning en la industria del automóvil

Autor: Pablo Lázaro Enguita
Director: Francisco Javier Rodríguez Sánchez

Tribunal:

Presidente: Pedro Martín Sánchez

Vocal 1: J.Antonio Jiménez Calvo

Vocal 2: Fco.Javier Rodriguez Sánchez

Calificación: .....
Fecha: .....

i. AGRADECIMIENTOS

En primer lugar, me gustaría dar las gracias a Francisco, mi tutor, por depositar su confianza en mí, para la realización de este TFG, tanto por guiarme, como por la ayuda prestada durante estos meses que ha durado la realización del mismo y atenderme en su despacho las veces que lo he necesitado y adaptándose a mi horario disponible.

De la misma manera, quiero agradecer a mis compañeros de empresa, los cuáles confiaron en mí desde el principio para llevar a cabo este proyecto y a la vez me prestaron una gran ayuda, sin la cual habría sido prácticamente imposible finalizarlo de manera satisfactoria y me han permitido adquirir innumerables conocimientos sobre este nuevo campo descubierto. También quiere agradecer a la empresa con la cual se ha realizado el proyecto ya que me han prestado sus conocimientos de mecánica y me han dado toda la información que he necesitado prácticamente en tiempo real y con una atención óptima

No quiero olvidarme de aquellos que empezaron la carrera siendo compañeros y dada la gran cantidad de horas que he pasado con ellos, la gran cantidad de retos y desafíos que he pasado junto a ellos se han convertido en amigos tanto en la universidad como en la vida diaria.

Por último, agradezco infinitamente a mi familia el apoyo recibido en todo momento, en especial a mis padres, mi hermano y Marta, los cuáles han contribuido a no rendirme en ningún momento y a esforzarme en la búsqueda de soluciones cuando el momento era delicado. Además gracias a ellos me he formado tanto académicamente como personalmente y sin ellos habría sido muy difícil llegar hasta aquí.

ii. RESUMEN

Hoy en día, las grandes instalaciones industriales tienen la necesidad de realizar acciones de mantenimiento predictivo sobre sus equipos haciendo uso de información sobre el estado de deterioro de sus elementos útiles. Gracias a este mantenimiento predictivo basado en aprendizaje automático o de máquina, es posible conocer la evolución del ciclo de vida de cierto elemento de la instalación mediante el exhaustivo análisis de magnitudes y campos altamente relacionados con su estado de degradación y rotura. Partiendo de un gran conocimiento del funcionamiento de los equipos y de aquellos comportamientos considerados anómalos, se consigue predecir de manera precisa el momento en el que se deben llevar a cabo acciones sobre esos determinados equipos realizando la sustitución del útil en cuestión y por tanto realizando, por ejemplo, acciones de mantenimiento.

En este Trabajo Fin de Grado se presenta una experiencia real en la implantación de algoritmos de aprendizaje automático para tratar de realizar mantenimiento predictivo sobre las pinzas robóticas que se encuentran en las líneas de producción de SEAT.

PALABRAS CLAVE

Mantenimiento predictivo, aprendizaje automático, pinzas robóticas, rozamiento datos.

iii. ABSTRACT

Nowadays In large industrial facilities, machine maintaining actions, based on its components degradation state, have crucial influence over production performance. Predictive maintenance systems target is acknowledging life cycle temporal evolution of certain element, through the analysis of magnitudes and fields high related with its degradation state. Accurate information about machine state enables us to determine when the friction is too high for the gripper of the robot.

During this study and in this document, it presents a real experience with the implementation of machine learning algorithms to try to perform a predictive maintenance on the robotic grippers that are in the production lines of SEAT.

KEYWORD

Machine learning, predictive maintenance, robot's gripper, friction, data.

iv. RESUMEN EXTENDIDO

La industria 4.0 ha permitido hacer de las grandes instalaciones industriales, plantas inteligentes gracias al procesamiento de la información de una manera eficiente. En primer lugar, la instalación debe poseer un alto grado de automatización, unida a una gran flexibilidad que permita aprovechar las futuras mejoras. Sin embargo, la alta complejidad, la automatización y la flexibilidad de una instalación inteligente exigen un alto control y seguridad para evitar problemas mayores.

Este documento presenta soluciones para realizar mantenimiento predictivo de las pinzas robóticas presentes en la línea de producción de una planta industrial del sector del automóvil, con la finalidad de conocer con exactitud su vida útil y las distintas tendencias a lo largo del tiempo de la variable objetivo conforme a su uso y deterioro progresivo.

Para ello el sistema de control del robot ubicado en la línea de producción, almacena en una carpeta compartida, archivos en formato Microsoft Access, que contienen todas las características mecánicas de la pinza durante la fase de fresado o mecanizado del vehículo durante su fabricación. Esta es, por tanto, la fase de *adquisición de datos* (escribe en diferente archivo cada día). Estos archivos requieren un tratamiento específico y un conocimiento acerca de las variables que intervienen en la rotura de la pinza y aquellos campos que están relacionados con la variable objetivo. Dada la gran cantidad de datos que se obtienen (cada vehículo pasa por un robot cada minuto), se ha investigado que instalación es la más problemática en cuanto a la rotura de la pinza, en este caso, debida al rozamiento. Para ello se ha utilizado una aplicación, *Kibana*, explicada más tarde. Una aplicación muy versatilidad que ofrece la posibilidad de graficar y soportar una gran cantidad de datos. Una primera problemática se presenta en el formato de los datos, ya que esta aplicación trabaja con formato JSON (*Javascript Object Notation*), mientras que los datos son escritos por el sistema de control del robot en .mdb (formato Microsoft Access). Por tanto, es necesario un tratamiento previo para cambiar el formato de los datos y posteriormente realizar la subida de estos datos a la aplicación *Kibana* para conseguir una visión analítica que permita determinar aquella instalación con un mayor coste de mantenimiento debido a la gran cantidad de roturas que se producen en sus pinzas. En el caso de este TFG se trata, como se observará más delante de la pinza robótica encargada de realizar fresados sobre el montante c de los vehículos.

A continuación, se ha utilizado el algoritmo que mejor se adapta al problema planteado en este TFG un tratamiento rápido y sencillo de los datos, lo que derivó en la utilización de la regresión lineal mediante mínimos cuadrados como método escogido.

Para comprobar la fiabilidad y precisión del algoritmo se analizan tres errores que relacionan la variable real y la variable estimada Además se aplica el algoritmo a varios casos y en varias ventanas de tiempos diferentes para comprobar los pros y contras del algoritmo utilizado, así como su precisión y exactitud.

Para concluir, una vez se ha dado con el algoritmo óptimo, se programa el algoritmo para obtener un ciclo completo de la pinza, ya que su finalidad además de determinar el valor del rozamiento de la pinza en cada momento, es poder determinar cuándo ésta pinza está llegando al final de su vida útil y realizar sobre ella un mantenimiento predictivo, es decir, adelantarse a su rotura (por ejemplo, sustituir la pinza durante el fin de semana, cuando se realiza una parada, en lugar de tener que detener la instalación durante el periodo de fabricación) y beneficiar a la instalación de realizar un mantenimiento predictivo en lugar de uno preventivo, reduciendo costes y mejorando su eficiencia.

Por tanto, el objetivo de este TFG es optimizar la línea de producción de una planta industrial del sector del automóvil mediante la implementación de algoritmos de aprendizaje de máquina o automático destinados a mantenimiento predictivo.

v. ÍNDICE DE CONTENIDOS

1 DESCRIPCIÓN DEL PROBLEMA Y OBJETIVOS 18

1.1 Robótica en la industria del automóvil 18

1.1.1 Evolución del parque de robots en España 18

1.2 Fabricación de un automóvil 19

1.3 Mantenimiento en la industria 23

1.3.1 Historia del mantenimiento 23

1.3.2 Mantenimiento predictivo para mantenimiento en planta 24

1.3.3 Mantenimiento predictivo pinzas robóticas 26

1.4 Descripción del problema 27

1.5 Objetivos 28

2 APRENDIZAJE AUTOMÁTICO 30

2.1 Definición conceptual de aprendizaje automático (machine learning) 30

2.2 Aprendizaje supervisado 31

2.2.1 Formulación de problemas 32

2.2.2 Errores medidos en el algoritmo supervisado: 36

2.3 Aprendizaje no supervisado 37

3 TRATAMIENTO DE DATOS 39

3.1 Introducción al tratamiento de datos necesario para el caso de uso del TFG 39

3.2 Datos pinzas .mdb 39

3.3 Visualización de datos mediante ElasticSearch 41

3.3.1 Conversión formato *Elastic* mediante *Netbeans* usando Java 41

3.3.2 Subida de datos 42

4 MÉTODO Y ALGORITMO EMPLEADOS 47

4.1 Introducción y elección algoritmo 47

4.2 Regresión lineal 47

4.2.1 Mínimos cuadrados (Least Squares) 49

4.2.2 Mínimos cuadrados recursivos (Recursive Least Squares) 51

4.3 Resultados 56

4.3.1 Mínimos cuadrados 56

4.3.2 Mínimos cuadrados recursivos 63

5 CÓDIGO 76

6 CONCLUSIONES 90
7 MANUAL DE USUARIO 92
7.1 Guía de instalación 92
7.2 Manual de usuario 92
8 PRESUPUESTO 94
8.1 Costes hardware 94
8.2 Costes software 94
8.3 Costes mano de obra 95
8.4 Costes totales 95
8.5 Presupuesto contractual 95
8.6 Monto total del presupuesto 96
9 BIBLIOGRAFÍA Y REFERENCIAS 97

vi. ÍNDICE DE FIGURAS

Figura: 1. Evolución parque robots. 18
Figura: 2. Robótica en automoción. 19
Figura: 3. Corte de piezas metálicas 20
Figura: 4.Clasificación de piezas 20
Figura: 5.Chasis y carrocería 20
Figura: 6.Pintura 21
Figura: 7.Ensamblaje. 21
Figura: 8.Exterior. 21
Figura: 9.Interior. 22
Figura: 10.Verificación. 22
Figura: 11.Banco pruebas. 23
Figura: 12.Industria 4.0. 25
Figura: 13.Conexión de datos. 25
Figura: 14.Proceso Aprendizaje. 33
Figura: 15.Probabilidad. 33
Figura: 16.Árbol de decisión. 34
Figura: 17.Regresión lineal. 35
Figura: 18.Regresión logística. 35
Figura: 19.Soporte vectorial. 36
Figura: 20.Componentes principales. 38
Figura: 21.Robot de soldadura. 40
Figura: 22.Robot lectura de datos. 40
Figura: 23.Datos Access. 40
Figura: 24.Curl subida de datos. 43
Figura: 25.Disposición datos Kibana. 43
Figura: 26.Dashboard pinzas. 44
Figura: 27.Esquema automóvil. 45
Figura: 28.Ejemplo de log pinza. 45
Figura: 29.Fn. de coste en regresión lineal. 49
Figura: 30.Caso 1. Predicción. 57
Figura: 31.Errores Caso 1. 57
Figura: 32.Caso 2. Predicción. 58
Figura: 33.Errores Caso 2. 58
Figura: 34.Caso 3. Predicción. 59
Figura: 35.Errores Caso 3. 60
Figura: 36.Ciclo pinza. Predicción. 62
Figura: 37.Errores Ciclo pinza. 63
Figura: 38.Caso 1 mínimos cuadrados recursivos λ=0.95. 64
Figura: 39.Errores Caso 1 mínimos cuadrados recursivos λ=0.95. 64
Figura: 40. Caso 1 mínimos cuadrados recursivos λ=0.90. 65
Figura: 41. Errores Caso 1 mínimos cuadrados recursivos λ=0.90. 66

Figura: 42. Caso 1 mínimos cuadrados recursivos λ=0.85. 66
Figura: 43. Errores Caso 1 mínimos cuadrados recursivos λ=0.85. 67
Figura: 44. Caso 2 mínimos cuadrados recursivos λ=0.95. 68
Figura: 45. Errores Caso 2 mínimos cuadrados recursivos λ=0.95. 68
Figura: 46. Caso 2 mínimos cuadrados recursivos λ=0.90. 69
Figura: 47. Errores Caso 2 mínimos cuadrados recursivos λ=0.90. 69
Figura: 48. Caso 2 mínimos cuadrados recursivos λ=0.85. 70
Figura: 49. Errores Caso 2 mínimos cuadrados recursivos λ=0.85. 70
Figura: 50. Caso 3 mínimos cuadrados recursivos λ=0.95. 71
Figura: 51. Errores Caso 3 mínimos cuadrados recursivos λ=0.95. 72
Figura: 52.. Caso 3 mínimos cuadrados recursivos λ=0.90. 72
Figura: 53. Errores Caso 3 mínimos cuadrados recursivos λ=0.90. 73
Figura: 54. Caso 3 mínimos cuadrados recursivos λ=0.85. 73
Figura: 55. Errores Caso 3 mínimos cuadrados recursivos λ=0.85. 74

vii. ÍNDICE DE TABLAS

Tabla 1.Errores Caso 1. 57
Tabla 2. Errores Caso 2. 59
Tabla 3. Errores Caso 3. 60
Tabla 4. Caso 1 Errores recursivos. 67
Tabla 5.Caso 2 Errores recursivos. 71
Tabla 6.Caso 3 Errores recursivos. 74
Tabla 7. Costes hardware. 94
Tabla 8.Costes software. 94
Tabla 9.Costes mano de obra. 95
Tabla 10.Costes totales. 95
Tabla 11.Monto total. 96
Tabla 12.Monto total con IVA. 96

Capítulo 1

1 DESCRIPCIÓN DEL PROBLEMA Y OBJETIVOS

1.1 Robótica en la industria del automóvil

El sector de la automoción en España acapara casi seis de cada diez robots, lo que supone en torno al 55% del parque, liderando la clasificación por encima de sectores como el sector de alimentos y bebidas o el del metal. De hecho, según datos de la Asociación Española de Robótica y Automatización de Tecnologías de la Producción (AER-ATP), el parque de robots en España en el 2016, última fecha de la que se tienen estudios, se compone de 34.528 unidades.

1.1.1 Evolución del parque de robots en España

A continuación, se muestra en la figura 1, un gráfico acerca de la evolución del parque robótico:

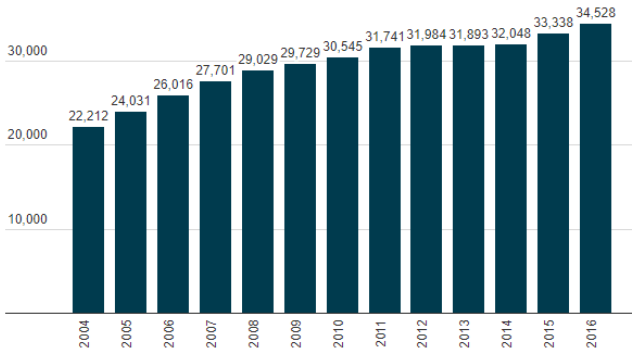


Figura: 1. Evolución parque robots.

En la industria de la automoción se emplean 19.022 robots según ha detallado la Asociación Española de Fabricantes de Automóviles Turismos y Camiones (Anfac). Esta cifra, que aumenta en más de 1500 unidades cada año, incluye cualquier actividad en el sector, desde fabricación hasta comercialización. A continuación, se muestra en la figura 2 la evolución del parque de robots en España de manera anual y por separado, indicando cuantos se usan en la industria del automóvil, y en otro tipo de industrias.

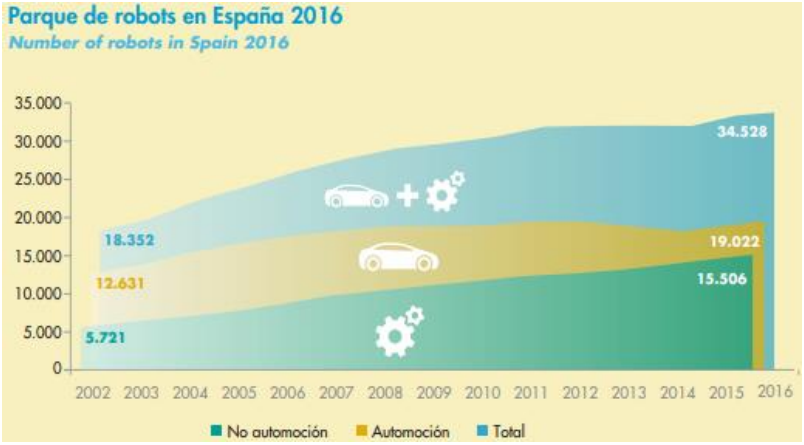


Figura: 2. Robótica en automoción.

La robotización de la automoción permite aumentar la calidad de los productos y la productividad. La resistencia y fiabilidad de los robots, combinada con la flexibilidad del ser humano, forman un binomio complementario. Por ello, la robotización no ha supuesto un gran impacto en el empleo.

Un 57,1% de los robots de automoción se dedican a tareas de carga y descarga, un 19,3 % a tareas de soldadura y fresado, que es la sección en la que se ubica este TFG, un 3,6 %para montaje y desmontaje y el restante en materiales.

1.2 Fabricación de un automóvil

Para poder ubicar dónde es factible realizar el mantenimiento predictivo en la automoción, primero se deben conocer brevemente las etapas de fabricación de un automóvil. La cadena de montaje consta de las etapas descritas a continuación (nos centraremos en las primeras donde apenas participación humana):

1. Corte de piezas metálicas: las piezas de chapa planas reciben el estampado del dibujo, después se corta el exceso de metal, se les da curvatura y se perforan los agujeros necesarios para su posterior ensamblado, tal y como se puede ver en la figura 3:

Figura: 3. Corte de piezas metálicas

2. Clasificación de piezas: como se puede apreciar en la figura 4, aquí se reciben y clasifican los distintos componentes para su posterior distribución a las zonas de la cadena de montaje que les corresponda.

Figura: 4.Clasificación de piezas

3. Chasis y carrocería: en el área de soldadura, los robots ensamblan las piezas de la carrocería, el chasis y la plataforma para construir el cuerpo del futuro vehículo como se puede comprobar en la figura 5.

Figura: 5.Chasis y carrocería

4. Pintura: en el horno de pintura, la carrocería se sumerge en una solución química que prepara el acero de la carrocería para recibir la pintura y se aplica una capa anticorrosiva en el exterior y en el interior. Después, se sellan los paneles y los robots suministran la capa de color definitiva en un proceso que dura unas 10 horas en total. A continuación, en la figura 6 se muestra una instantánea de un vehículo en la zona de pintura:

Figura: 6.Pintura

5. Ensamblaje mecánico: antes de introducir el conjunto mecánico se retiran las puertas para facilitar el resto del montaje. Luego, un elevador coloca el motor en su espacio para que sea atornillado. En esta fase también se instalan las suspensiones, el cableado, el sistema de escape, los ejes, la transmisión y la columna de dirección. En la figura 7 se puede apreciar a un operario en esta zona:

Figura: 7.Ensamblaje. .

6. Exterior: con el conjunto mecánico ya instalado, comienza la fase final de la cadena de montaje que en su primera etapa consiste en «vestir» el cuerpo del coche. Primero se procede a ensamblar las piezas plásticas que completan la carrocería, como los paragolpes. También se instalan los conjuntos lumínicos y las ruedas. El exterior del vehículo se completa con la fijación de las lunas trasera y delantera. En la figura 8 se muestra al vehículo siendo mecanizado exteriormente:

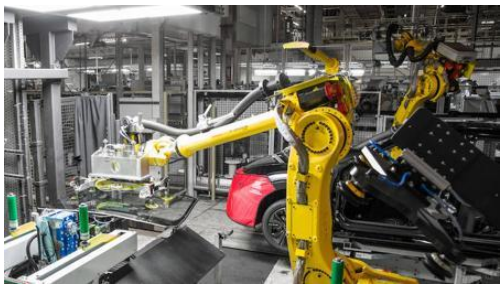


Figura: 8.Exterior.

7. Interior: en esta fase, los operarios completan el acabado interior con asientos, salpicadero, volante, cinturones, etc. Para terminar con la reinstalación de las puertas como se puede apreciar en la figura 9:

Figura: 9.Interior.

8. Verificación: antes de abandonar la cadena de montaje, cada vehículo recibe una inspección final, como se puede visualizar en la figura 10, por parte de los trabajadores, para comprobar que todo esté en orden. El último paso es inyectar los fluidos necesarios para realizar las pruebas mecánicas.

Figura: 10.Verificación.

9. Banco de pruebas: a cada coche se le realiza una valoración eléctrica y mecánica, la prueba de lluvia, para detectar posibles entradas de agua, y, en el circuito, se comprueban los frenos y la respuesta de amortiguadores y la dirección en distintas superficies. En la figura 11 se observa el vehículo completamente acabado y a falta únicamente de las pruebas oportunas.

Figura: 11.Banco pruebas.

Para este TFG se ha dispuesto de datos de varias instalaciones dentro de la cadena de montaje principalmente de la etapa 1 y 3, en lo referido a corte de piezas metálicas y al chasis y carrocería.

1.3 Mantenimiento en la industria

1.3.1 Historia del mantenimiento

El desarrollo del mantenimiento industrial está ligado a la evolución Técnico-Industrial de la humanidad. Hasta 1914 eran acciones consideradas de importancia secundaria, hasta que la implantación de la producción en serie por la compañía Ford Motor, generó la necesidad de constituir unidades para llevar a cabo el mantenimiento de las máquinas de la línea de producción en el menor tiempo posible, asegurando siempre una planificación y un mínimo de interrupción de la producción. Pero todavía en este momento el mantenimiento se consideraba una actividad meramente correctiva y preventiva, no se entendía la posibilidad de un mantenimiento predictivo.

Posteriormente, durante la Segunda Guerra Mundial, a consecuencia del desarrollo de la industria militar, aparece el concepto de mantenimiento preventivo motivado por el aumento de la mecanización y la obligación de reducir los plazos de entrega.

Más tarde, en la década de los 60, comenzaron a automatizarse las industrias de forma generalizada, verificándose la ley de “a más máquinas mayor número de fallos”. Durante los últimos veinte años el mantenimiento ha evolucionado, desde el punto de vista de la gestión, más que cualquier otra disciplina gerencial. El aumento de recursos en esta época para tener superioridad sobre el rival permitió hacer esfuerzos económicos para las acciones de mantenimiento que desarrollaron técnicas de planificación y control.

De hecho, en la actualidad se requiere la máxima disponibilidad de las instalaciones a la vez que se plantea su eficiencia, es decir, minimización de los costes derivados de la Gestión del Mantenimiento. A esto se suman los nuevos criterios impuestos por los

gobiernos destinados a proteger el medio ambiente y de dotar de una mayor seguridad a las instalaciones industriales de cara a evitar catástrofes y evitar pérdidas humanas.

Por todos estos motivos, deben establecerse procedimientos y técnicas propias de la ingeniería que optimicen los recursos que se destinan a la función de mantener operativas las instalaciones.

1.3.2 Mantenimiento predictivo para mantenimiento en planta

El mantenimiento predictivo es uno de los campos de mayor potencial de las técnicas de aprendizaje automático dentro de la industria 4.0 (figura 12). La tendencia en las actividades de mantenimiento ya sea de infraestructuras o de sistemas, se está orientando hacia un mantenimiento según estado, si bien este tipo de mantenimiento supone una gran mejora en cuanto a costes, su éxito depende de la monitorización o supervisión continua de los elementos a mantener para conocer su estado.

Este tipo de mantenimiento permite determinar cuándo un elemento va a requerir una actuación en función de su estado, pero no es posible conocer cuándo se van a dar esas condiciones ni cómo afectan en su vida útil.

Uno de los principales objetivos de la industria hoy en día, es optimizar al máximo el rendimiento, la disponibilidad y los activos en las plantas de fabricación. Para ello son adecuadas las técnicas de aprendizaje automático que permiten optimizar los modelos de mantenimiento y operación.

El mantenimiento preventivo es aquel en el cual se realiza periódicamente la sustitución de componentes de los equipos basándose en el estudio del comportamiento de cada componente. Esta técnica se centra en la duración de la vida útil de cada componente en el tiempo, sin que importe las causas que provocaron el fallo. Un ejemplo donde se realiza un mantenimiento preventivo es el cambio del aceite del motor de un automóvil. Cada cierta cantidad de kilómetros recorridos se sustituye el aceite usado por uno nuevo para prevenir fallos mayores en el motor.

La conexión de las máquinas industriales a internet ha permitido pasar de un *mantenimiento reactivo o preventivo* a un *mantenimiento predictivo* mucho más exhaustivo que ayuda a anticiparse a los problemas. El mantenimiento reactivo, el habitual hasta ahora, se basa en actuar sobre la máquina o pieza averiada en el momento en el que comienzan a producirse lo fallos, lo que provoca un aumento de costes, una reducción en la productividad y, sobre todo, de la fiabilidad de la máquina,lo que lleva consigo un aumento de costes laborables.

Figura: 12.Industria 4.0.

Esta clase de analítica avanzada es clave en el marco de la fábrica inteligente del futuro, a fin de identificar la degradación del sistema antes de que se produzca cualquier error. Los fallos inesperados son uno de los principales factores en los costos de mantenimiento, pero contar con capacidades predictivas no sólo reduce costes sino, tal y como se ha comentado con anterioridad, disminuye tiempos de inactividad e incrementa la productividad.

Además, la digitalización de los procesos de mantenimiento actuales (figura 13) están generando gran cantidad de información, que convenientemente tratada podría desvelar patologías y comportamientos que en la actualidad están ocultos y que pueden aportar otros enfoques muy útiles en el ámbito del mantenimiento predictivo sobre las plantas industriales.

Figura: 13.Conexión de datos.

Las ventajas de realizar un mantenimiento preventivo son:

- Incremento de la vida útil y del rendimiento de las instalaciones.

- Mejora la optimización de los recursos y desmontajes innecesarios.

- Mejora de la calidad del producto fabricado.

- Reduce los stocks de material de mantenimiento.

- Mayor seguridad laboral.

- Se limitan prácticamente en su totalidad las penalizaciones por retrasos en las entregas.

- Generación de una base de datos para posibles estudios.

-Mejora del conocimiento sobre el funcionamiento de las máquinas y el sistema productivo.

- Ahorro de entorno al 60% del coste de manteniendo en comparación con el mantenimiento correctivo (aquel que corrige los defectos observados en los equipamientos o instalaciones, y consiste en localizar averías o defectos y corregirlos o repararlos.)

-Las técnicas de monitorización pueden aplicarse de forma automática, pudiendo realizarse el control desde centros distanciados de la maquinaria.

Los *inconvenientes* de realizar un mantenimiento preventivo son:

- Sustitución de elementos con anterioridad al fin de su vida útil.

- Necesidad de realización de una buena planificación.

- Requiere de la experiencia del personal para realizar la planificación y una mayor formación del personal de mantenimiento, ya que necesitan conocer las técnicas

- Genera gastos fijos.

1.3.3 Mantenimiento predictivo pinzas robóticas

En el caso de este TFG, se dispone de datos obtenidos durante 2 meses sobre las pinzas de los robots de las distintas instalaciones de fabricación de automóviles en una empresa de primer nivel. Las pinzas analizadas tienen como principal función soldadura y fresados. Cabe destacar que las instalaciones detienen la fabricación de automóviles todos los fines de semana, así como aquellos días festivos tanto a nivel nacional como a nivel autonómico; por ello solo se dispone de datos de lunes a viernes de cada semana desde las 00:00 del lunes hasta las 23:59 del viernes de esa misma semana.

La variable a tener en cuenta en la pinza es el **RozU** (*rozamiento usado*) que debe mantenerse siempre oscilando entre -0.4 y 0.4. Existen valores negativos dado que en función del sentido de trabajo de la pinza, este valor será positivo o negativo. Una vez el valor se sale de esos rangos de una manera continuada la pinza se rompe.

El *rozamiento usado* de la pinza viene determinado por otras 4 variables que, bien por el paso del tiempo o bien en las arrancadas tras alguna parada por rutina o avería de

alguna instalación, sufren algunos desajustes. Estas variables que intervienen directamente en el **RozU** son la posición del electrodo fijo (**DespElFj**) respecto a la posición del electrodo de chapa (**DElCh**) y el grosor de chapa medido (**GrChM**) respecto al grosor de chapa programado (**GrCh**).

En concreto la posición del electrodo fijo no debe exceder en 3 unidades la posición del electrodo de chapa, en ese caso la pinza experimenta un mayor rozamiento hasta producir su rotura superando el 0.4 de rozamiento límite. Algo semejante ocurre con el grosor de chapa, ya que el grosor de chapa medido nunca debe sobrepasar el grosor de chapa programado, en ese caso el rozamiento de la pinza también se ve afectado considerablemente.

En este caso se disponen de 4 variables o campos, dos de los cuales estás interrelacionadas por lo que se deberá realizar un tratamiento previo de los datos para reducir la dimensionalidad del problema de aprendizaje automático planteado.

Gracias a estos campos y la relación con la variable problemática se conseguirá establecer un modelo que nos permita realizar un mantenimiento predictivo sobre las pinzas robóticas de esta instalación automovilística, y conocer el momento en el que deberá ser cambiada esa pinza, aprovechando al máximo su vida útil.

1.4 Descripción del problema

El problema a tratar tiene varias fases claramente diferenciadas, en primer lugar, es necesario conocer el funcionamiento del aprendizaje automático desde sus fundamentos hasta los métodos que mejor se adapten al problema planteado en este TFG.

El punto de partida de todo el proceso pasa por adquirir los datos necesarios para poder sacar alguna conclusión futura. Los datos adquiridos, en el caso de este TFG, se almacenan en archivos diarios que posteriormente se podrán analizar.

Una vez adquiridos los datos necesarios, hay que adaptar el formato con el que el sistema de control del robot los almacena para adaptarlos al formato que requiere la aplicación de análisis y representación. Por tanto, surge la necesidad de realizar un exhaustivo tratamiento de datos cambiando su formato, mediante un proceso capaz de manejar la gran cantidad de datos generados, resultado de la automatización de una instalación de enromes dimensiones, con gran cantidad de robots y pinzas.

Tras realizar este tratamiento de datos y posteriormente la aquella instalación que posee el robot con la pinza más problemática durante la fabricación del vehículo, se procede a la elección y programación del algoritmo de aprendizaje automáticocque mejor se adapte a la finalidad de detectar y analizar los valores de la variable objetivo

que constituye la salida, RozU (rozamiento usado), a partir de las variables de entrada (aquellas que influyen de una manera decisiva sobre la variable objetivo).

Para poder analizar si las predicciones hechas son correctas, se pondrá el modelo a prueba y se calcularán los errores que indica cómo de exactas y precisas son las estimaciones realizadas. Para ello se dispondrá de un conjunto de datos de prueba y uno de entrenamiento.

Por último, se comprobará si el algoritmo cumple con su objetivo principal, detectar la etapa en la que la pinza robótica puede romper y debe sustituirse por otra nueva sin afectar al rendimiento de la instalación, estimando los valores con el algoritmo programado de un ciclo de pinza completo.

1.5 Objetivos

Se muestran en orden cronológico, tal y como se han ido alcanzando:

1. Revisión de conocimientos sobre aprendizaje automático para poder desarrollar técnicas útiles en el caso de uso propuesto.
2. Tratamiento de datos para adaptar formatos entre los ficheros de salida del sistema robótico y el formato de entrada requerido por la aplicación de análisis.
3. Introducción a la herramienta Elastic (Kibana) y al lenguaje de programación Java mediante la plataforma Netbeans para convertir los archivos Microsoft Access a JSON.
4. Estudio y visualización de los datos para centrar el estudio en la pinza del robot de la instalación que resulte más problemática.
5. Aplicación de algoritmos de aprendizaje automático utilizando regresiones con mínimos cuadrados en Matlab, obteniendo conclusiones definitivas acerca de la duración y la evolución del ciclo de la pinza con un error minimizado dentro de las limitaciones.

Capítulo 2

2 APRENDIZAJE AUTOMÁTICO

2.1 Definición conceptual de aprendizaje automático (machine learning)

Una definición relativamente general de **aprendizaje** dentro del contexto humano podría ser la siguiente: "*proceso a través del cual se adquieren o modifican habilidades, destrezas, conocimientos, conductas o valores como resultado del estudio, la experiencia, la instrucción, el razonamiento y la observación*". No se considera aprendizaje aquello innato, sino aquello que procede de la experiencia con el entorno.

El Machine Learning o aprendizaje automático es una disciplina científica dentro de la inteligencia artificial cuyo objetivo es programar algoritmos para que los sistemas sean capaces de aprender o desarrollar comportamientos a partir de información obtenida en ejemplos anteriores. El objetivo prioritario es usar la evidencia conocida, mediante la creación de una hipótesis y pudiendo así responder ante situaciones a priori, desconocidas. Se trata de un campo que también se centra en el estudio de la complejidad computacional de los problemas, y guarda bastante relación con la estadística y las matemáticas.

El aprendizaje automático se podría resumir en una frase dada por la gran parte de especialistas en el sector:

"*Se dice que un programa de computación aprende de la experiencia E con respecto a una tarea T y alguna medida de rendimiento P, si es que el rendimiento en T, medido por P, mejora con la experiencia E*".

En resumen, el objetivo del programador es crear un programa que pueda aprender a realizar una tarea, y mejorar su rendimiento en la misma mediante entrenamiento (experiencia).

Las aplicaciones donde hoy en día se utiliza esta técnica son muy diversas:

- Seguridad de datos: el *malware* es un problema creciente hoy en día. Pese a que casi a diario surgen nuevos malware, suelen tener casi el mismo código que las versiones anteriores. De esta manera, el aprendizaje automático puede predecir qué archivos son malware con gran precisión**.** Es más, los algoritmos de

aprendizaje automático pueden buscar patrones en cómo se accede a los datos en la nube e informar de anomalías que podrían predecir infracciones de seguridad.

- Búsquedas online: este campo es el más utilizado. Empresas como Google, Facebook y sus competidores, mejoran continuamente el motor de búsqueda basándose en la información y el tiempo empleado por el usuario en determinadas búsquedas y campos.

- Medicina: campo de gran utilidad en diagnósticos, partiendo del historial clínico y síntomas del paciente.

- Comercio financiero: muchas empresas del sector ya utilizan sistemas propios para predecir y ejecutar operaciones de gran volumen a altas velocidades, con un análisis masivo de datos inalcanzable para el ser humano.

- Procesamiento del lenguaje natural (NLP): estos algoritmos pueden suplir al ser humano en sectores de atención al cliente y dirigir a los usuarios a la información que necesitan.

- Coches inteligentes: este prometedor sector aprendería sobre el propietario del vehículo y su entorno, ajustando la configuración interna (temperatura, audio, posición del volante y asiento…).

- Juegos: en el ámbito del ocio, el aprendizaje automático, se utiliza para dar inteligencia a los enemigos a los que tenga que enfrentarse el usuario.

Dentro de Machine Learning se pueden incluir técnicas de Aprendizaje Supervisado (las empleadas en el trabajo actual), y No Supervisado

2.2 Aprendizaje supervisado

En el aprendizaje supervisado, los algoritmos trabajan con datos "etiquetados" (*labeled data*), intentando encontrar una función que, dadas las variables de entrada (input data), les asigne la etiqueta de salida adecuada. El algoritmo usado se entrena con un "histórico" de datos y así "aprende" a asignar la etiqueta de salida adecuada a un nuevo valor de entrada, es decir, actúa prediciendo el valor de salida. Por tanto, se trata de un proceso de aprendizaje donde el entrenamiento se controla por un agente externo, realizándose predicciones basadas en datos históricos almacenados.

Por ejemplo, disponiendo de información de viviendas (el conjunto de ejemplos), y los precios correspondientes a cada una de ellas (respuesta), se trata de aprender alguna regla de correspondencia entre esas viviendas y sus respectivos precios, que permita predecir el coste, de manera fiable, cuando tengamos información de nuevas viviendas.

2.2.1 Formulación de problemas

X– conjunto de objetos.

Y – conjunto de etiquetas (o respuestas).

$y: X \rightarrow Y$ – función de dependencia entre los objetos y etiquetas.

$y_i = y(x^{(i)}), i = 1, \dots l$ – valores conocidos de la función.

Ahora la **problemática** es encontrar una función h $h: X \rightarrow Y$, que permita aproximar el conjunto de objetos al de respuestas. Se utiliza un algoritmo de aprendizaje, que permite aproximar esta función.

También intervienen una serie de *predictores* conocidos como características de dicho objeto, en el caso de las viviendas podrían ser el tamaño, la antigüedad, zona de la ciudad, número de pisos, etc…., es decir, todas aquellas características que tienen influencia en el precio de la vivienda.

Cada objeto se representa como un vector de n dimensiones $x^{(i)} \epsilon\ \mathbb{R}^n$, donde n es el número de **características**. Una etapa clave de la realización del algoritmo es seleccionar correctamente las características para un correcto desarrollo del algoritmo, así como su naturaleza.

Para este tipo de aprendizaje contamos con un conjunto de entrenamiento:

$$\{(x^{(i)}, y^{(i)}), \dots, (x^{(m)}, y^{(m)})\} \in X$$

Siendo:

(x_i, y_i), la pareja que representa a un objeto junto a su etiqueta, lo que se conoce como ejemplo de entrenamiento.

Este algoritmo permite aportar una aproximación a la función de dependencia desconocida entre objetos y respuestas. A continuación, se muestra en la figura 14 el proceso de aprendizaje:

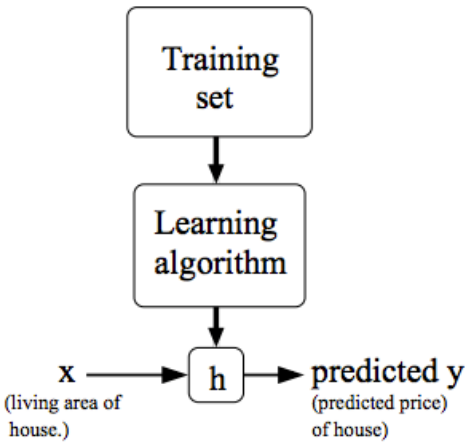
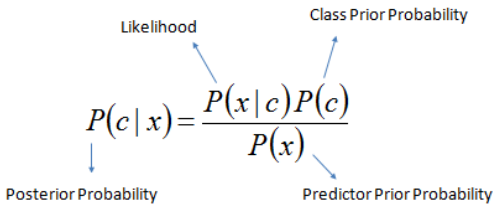


Figura: 14.Proceso Aprendizaje.

Existen muchos algoritmos de aprendizaje supervisado, entre los que son más importantes:

- Algoritmo de clasificación Naïve Bayes (figura 15): es una herramienta, que permite representar de forma estructurada la probabilidad conjunta de diferentes variables aleatorias, utilizando para ello un modelo gráfico de las relaciones existentes entre dichas variables. Los clasificadores Näive Bayes son una familia de clasificadores probabilísticos basados en la aplicación del teorema Bayes. Siendo P(c|x) la probabilidad a posteriori, P(x|c), probabilidad, P(c) probabilidad a priori de clase y P(x) la probabilidad a priori del predictor.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figura: 15.Probabilidad.

Aunque habitualmente son utilizados para inferir probabilísticamente los valores de algunas de las variables que contienen, también es posible realizar otro tipo de operaciones como encontrar la configuración de variables más probables, medir posibles conflictos entre los calores de las variables de la red o realizar análisis de sensibilidad sobre la influencia de unas variables en otras.

- Árboles de decisión: es una herramienta de apoyo a la decisión que utiliza un gráfico o un modelo similar a un árbol de decisiones y sus posibles consecuencias, incluidos los resultados de eventos fortuitos, los costes de recursos y la utilidad. La figura 16 muestra esta herramienta:

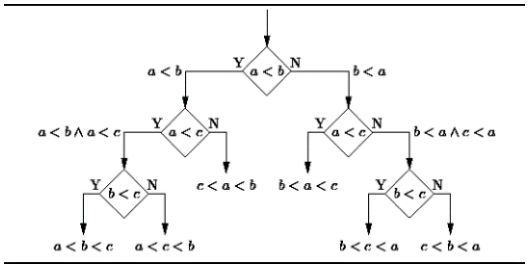


Figura: 16.Árbol de decisión.

Este algoritmo de clasificación representa un conjunto de restricciones o condiciones que se organizan de forma jerárquica y que se aplican sucesivamente desde una raíz hasta llegar a un nodo terminal u hoja del árbol. Este método se encuentra entre los más populares y es utilizado con éxito en diferentes aplicaciones, desde sistemas de diagnóstico médico hasta sistemas de determinación de crédito bancario. La estrategia está basada en ejemplos y presenta al sistema un conjunto de casos relevantes para clasificar y desarrollar un árbol de decisión baasado en el conocimiento, desde la raíz hasta las hojas sin importar el orden de llegadas de los casos. Pese a tratarse de sistemas de propósito general, las aplicaciones van dirigidas a procesos de clasificación.

- Modelos de regresión lineal: se trata de una técnica estadística utilizada para investigar la relación entre dos o más variables. Es un modelo matemático usado para aproximar la relación de dependencia entre una variable dependiente “Y” y las variables independientes “Xi”. El término lineal se emplea para distinguirlo del resto de técnicas de regresión que emplean modelos basados en cualquier clase de función matemática. Los modelos lineales son una explicación simplificada de la realidad, mucho más ágiles y con un soporte teórico mucho más extenso por parte de la matemática y la estadística. La primera idea es entenderlo como la tarea mediante la cual se puede ajustar una línea recta a través de un conjunto de puntos.

  Existen diversas estrategias para realizar esta tarea, uno de ellos es el ajuste por “mínimos cuadrados”: se dibuja una línea, y posteriormente para cada uno de los puntos de datos, se miden las distancias entre los puntos y la línea de regresión y se suman después de elevarlas al cuadrado. La línea ajustada sería aquella en la que esta suma de

distancias sea lo más pequeña posible. Se muestra un ejemplo a continuación en la figura 17:

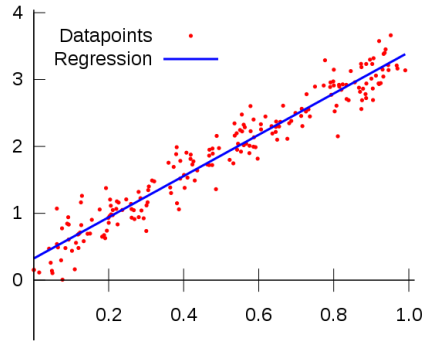


Figura: 17.Regresión lineal.

Esta técnica es muy utilizada para realizar predicciones de una variable (objetivo) en términos de otras (regresivas).

En función de las variables a investigar podemos tener regresión simple o regresión múltiple. Cuando la predicción de una variable se realiza únicamente en términos de una variable, hablamos de regresión simple. Por el contrario, cuando la predicción se hace teniendo en c

- Métodos de regresión logística (figura 18): este método mide la relación entre la variable dependiente categórica y una o más variables independientes estimando las probabilidades utilizando una función logística.

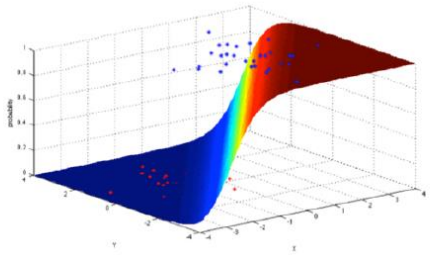


Figura: 18.Regresión logística.

- Máquinas de vectores de soporte (SVM): se trata de un algoritmo de clasificación binario, representado en la figura 19, en el cual, dado un conjunto de puntos de 2 tipos en el lugar N dimensional, SVM genera un hiperlano (N-1) dimensional para separar esos puntos en 2 grupos. Esta herramienta proporciona una línea recta que separa esos puntos en 2

tipos, es por ello que este algoritmo se engloba dentro de las técnicas de clasificación.

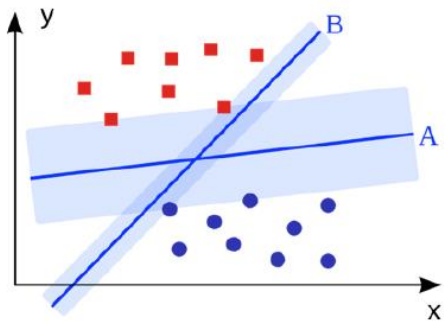


Figura: 19.Soporte vectorial.

Como podemos ver en la figura anterior, el aprendizaje supervisado está formado por tres pasos principalmente:

1. Se usa el conjunto de entrenamiento junto con un algoritmo que permita crear una hipótesis fiable.
2. a) Prueba, donde se utiliza la hipótesis obtenida en el paso anterior para realizar nuevas predicciones.
   b) Para realizar las predicciones es necesario medir su calidad, que vendrá dada por una serie de medidas del error de predicción.

2.2.2 Errores medidos en el algoritmo supervisado:

- ✓ **MAE** (mean absolute error o error medio absoluto)

  Es una medida de la diferencia entre dos variables continuas. Suponiendo que X e Y son variables de observaciones emparejadas que expresan el mismo fenómeno. Los ejemplos de Y frente a X incluyen comparaciones entre tiempo predicho y observado.

$$\text{MAE}=\frac{1}{n}\sum_{t=1}^{n}|e_t|$$

Donde $e_t=f_t - y_t$, siendo y_t el valor real y f_t el valor predicho.

- ✓ **MAPE** (mean absolute percentage error o error porcentual absoluto medio)

  Es una medida de la precisión de predicción de un método de pronóstico en estadística, por ejemplo, en la estimación de tendencias. Por lo general, expresa precisión como un porcentaje, y se define por la fórmula:

$$\text{MAPE}=\frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right|$$

- ✓ **RMSE** (root mean square error o error medio cuadrático).

  Es una medida estadística de la magnitud de una cantidad variable. Puede calcularse para una serie de valores discretos o para una función matemática de variable continua. El nombre deriva del hecho de que es la raíz cuadrada de la media aritmética de los cuadrados de los valores.

$$\text{RMSE}=\sqrt{\frac{1}{n}\sum_{t=1}^{n}|{e_t}^2|}$$

2.3 Aprendizaje no supervisado

Este método de aprendizaje automático usa un modelo que es ajustado a las observaciones. En contraposición al supervisado, no hay un conocimiento a priori. En este caso un conjunto de datos de objetos de entrada es tratado como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos.

También es útil para la compresión de datos: fundamentalmente, todos los algoritmos de compresión dependen tanto explícita como implícitamente de una distribución de probabilidad sobre un conjunto de entrada. Existen diversos métodos de agrupamiento representados en la figura 28:

- ➢ Algoritmos clustering: consiste en agrupar un conjunto de objetos tales que los objetos en el mismo grupo o cluster, son más similares entre sí que a los de otros grupos.
- ➢ Análisis de componentes principales: es un procedimiento estadístico que usa una transformación ortogonal para convertir un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores variables linealmente no correlacionadas llamadas componentes principales.

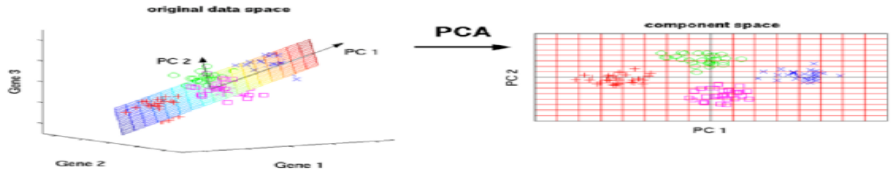


Figura: 20.Componentes principales.

Capítulo 3

3 TRATAMIENTO DE DATOS

3.1 Introducción al tratamiento de datos necesario para el caso de uso del TFG

Partiendo de la situación inicial, se tienen los datos escritos el sistema de control por los robots en una carpeta compartida. Como se ha comentado antes estos datos se encuentran formato Microsoft Office Access (.mdb). Por tanto, una vez tenemos adquiridos los datos, la primera tarea a realizar es tratarlos de manera adecuada para cambiar al formato que requiere la herramienta Elastic (Kibana).

Es muy importante que estos archivos estén de una manera estructurada, en este caso se encuentras en archivos diarios perfectamente estructurados en los que, como más adelante se verá cada columna conforma cada uno de los campos que tiene esa pinza determinada. Por tanto, se comienza de partida con el análisis de datos en el formato original para su posterior tratamiento.

3.2 Datos pinzas .mdb

En este TFG se tratan los datos de las pinzas de los robots de una de las principales fábricas de automoción de España, que realizan una serie de maniobras en el proceso de fabricación de sus automóviles, esos robots se encuentran en diferentes instalaciones, como por ejemplo el robot apreciado en la figura 14 (“MQB37 UB1 COMP PC16”, “MQB37 UB1 GEO”, “MQB37 PISO ANT PC15”, “MQB37 MASCA 3 Y 4”, “MQB37 SALPICADERO PC1” , ……) las cuales poseen diferentes robots en cada una de ellas. Cada robot posee numerosos brazos, y en sus extremos posee numerosas pinzas (“KASTWA110050R01EZ “, “KBSSIR030.2EZ-1 “, “KAAUF2320090R06-EZ1 “, “PES110R03_EZ1”, “KABHVS110060R01EZ-1”, …) con distintas funcionalidades, de hecho, cada una tiene más de un programa cargado ya que una misma pinza tiene diversos movimientos y funcionalidades.

Figura: 21.Robot de soldadura.

Figura: 22.Robot lectura de datos.

Estos robots escriben diariamente (representación metafórica en figura 22), en función del robot, en diferentes intervalos de tiempo, pero todos ellos en un rango pequeño. Cada día equivale a un archivo Microsoft Office Access (.mdb) con infinidad de características mecánicas y eléctricas de cada pinza de cada robot de cada instalación como se muestra en la figura 23:

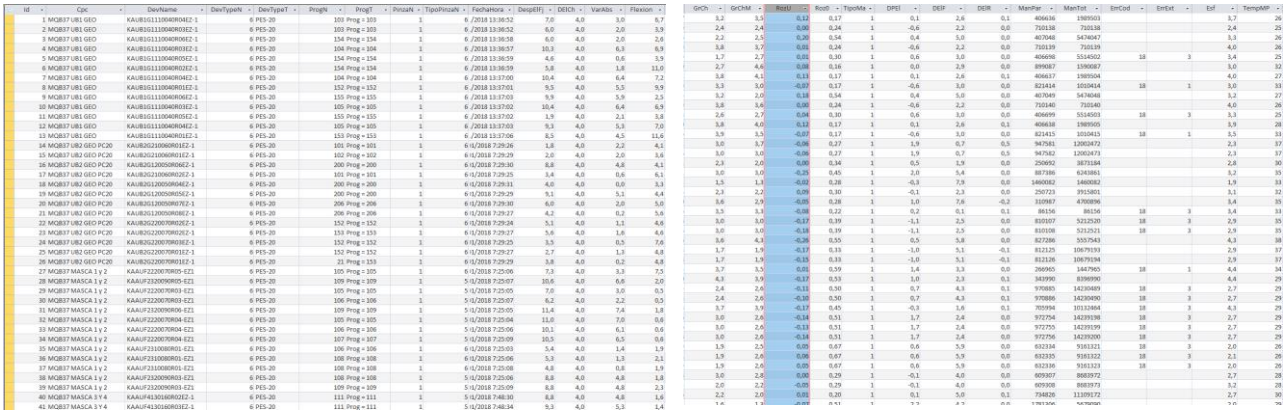


Figura: 23.Datos Access.

En sombreado se encuentra la variable que queremos como salida como explicaremos más adelante.

Se tienen datos de los dos primeros meses de 2018, con un total de más de 100 millones de datos. Este formato de datos (Access) debe adaptarse al requerido por la plataforma que se va a usar para identificar el problema y ver con claridad en qué robot es más útil aplicar las técnicas de mantenimiento predictivo.

Dada esta problemática se usará la herramienta Elastic para poder avanzar en el estudio.

3.3 Visualización de datos mediante ElasticSearch

Se trata de un servidor de búsqueda basado en *Lucene* (es una API de código abierto para recuperación de información, normalmente implementada en Java) útil en cualquier aplicación que requiera indexado y motores de búsqueda.

Esta herramienta requiere documentos *JSON* (**Java Script Object Notation**), un formato de texto ligero de intercambio de datos que se utiliza como alternativa al *XML*. Se utiliza este formato debido a la facilidad de desarrollo y posee una mayor polivalencia.

Por tanto, hay que transformar archivos .mdb en JSON (extensión necesaria para utilizar la herramienta de Elastic). Para ello se ha realizado una aplicación en Java mediante el programa **Netbeans**.

3.3.1 Conversión formato *Elastic* mediante *Netbeans* usando Java

En el capítulo 5 se adjunta el código empleado con algunas anotaciones, pero a continuación se explica brevemente su función.

Este proyecto Gradle (Java) incluye clases para lectura y escritura de archivos con unas determinadas características. Los ficheros de entrada deben ser siempre Microsoft Office Access (“.mdb” o “.accdb”) y los resultantes del procesamiento serán JSON (“. json”).

En primer lugar, lee MS Office Access de los subdirectorios de un directorio raíz. Procesa sus datos y genera los correspondientes archivos JSON para cada número de filas que se le indique en el código. A mayor número de filas los archivos comienzan a tener un gran tamaño lo que genera mayor dificultad para procesarlos y subirlos a la plataforma que se usará en un futuro. En ocasiones se extraen datos de bases de datos como MongoDB (un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto), en lugar de determinados directorios, en ese caso habría que indicar el nombre de la colección escogida en Mongo.

Argumentos que pasar en orden de ejecución:

```
/*
* arg[0] - Directory with all the the '.mdb' or '.accdb' files.
* arg[1] - Connection string to connect with the MongoDB database.
* arg[2] - Database name.
* arg[3] - Collection name.
*/
```

A continuación, se muestra cómo se indica el directorio, y el lugar en el que se programa el número de líneas:

```
public static void main(String args[]) throws Exception {
    args = new String[]{"C:\\Users\\plazaro\\Desktop\\Maniobras", "pinzas-maniobras", "doc"};
    File dir = new File(args[0]);

    if (dir.isDirectory()) {

        for (File f : dir.listFiles()) {
            new AccessReaderManiobrasJSON(f.getAbsolutePath()).generateJSONFile(args[1], args[2]);
        }

    }
```

```
if (i % 9999 == 0 || i == (table.getRowCount() - 1)) {
    jsonFile = new File(jsonPath + i + ".json");
    fw = new FileWriter(jsonFile);

    /* File exporting. */
    fw.write(jsonText);

    /* File writer closure. */
    fw.close();

    jsonText = "";
}
```

En resumen, la misión del código es en primer lugar extraer los archivos en formato ".mdb" o ".accdb" desde una base de datos o un directorio, para leerlos y escribirlos en formato JSON proporcionándoles el formato (string, bool, char, float) y la representación adecuada; notación científica, puntos en lugar de comas, la fecha y la hora en el formato adecuado…En el capítulo 5 aparece el código en el cual se encuentran los 2 archivos java para generar los JSON con anotaciones sobre la función de cada fragmento de código.

Una vez se tienen los archivos JSON en la carpeta indicada se procede a subir los archivos al **logstash** de Elastic, una herramienta para la administración de logs, que ofrece la posibilidad de recolectar, parsear y guardar los logs para futuras búsquedas.

3.3.2 Subida de datos

Para proceder a la subida de datos masiva desde local, se ha usado *curl* (herramienta para transferir archivos desde un servidor). Se abre una consola de comandos y se abre la carpeta en la que están los JSON con los datos y sus comandos.

Es muy importante el formato de estos y debe seguir la siguiente estructura:

```
{"index" : { "_index" : "robots-kr210r2700-sollposition", "_type" : "doc"}}
{"Zeit":1515775017.628,"Sollposition":-4721.3055}
{"index": { "_index" : "robots-kr210r2700-sollposition", "_type" : "doc"}}
{"Zeit":1515775017.632,"Sollposition": -4721.3055}
```

Una primera línea en la que definiremos a que índice pertenece, es decir donde se tiene que insertar y una segunda línea con los datos. Cabe destacar que tantos datos como acciones han de estar en una única línea dado que por debajo se utilizan los retornos de carro para saber dónde finalizan los datos. Además, como se puede apreciar no van separadas por comas, y en la última línea es necesario añadir un retorno de carro para terminar la inserción.

Una vez están los JSON preparados y el *cmder (consola de comandos)* ubicado en la carpeta (directorio) adecuada ejecutamos el siguiente comando en *bash,* código repsentado en la figura 24 (se trata de un programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de consola):

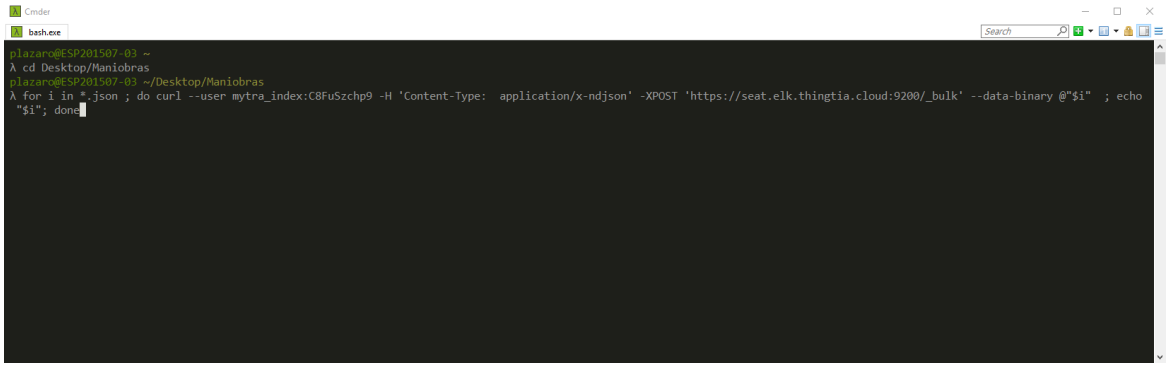


Figura: 24.Curl subida de datos.

Una vez subidos todos los datos procedemos a entrar en **Kibana**, herramienta de Elastic (la cual posee un nodo asociado a un *logstash* donde se guardan todos los datos con los que es posible trabajar en Kibana) en la cual se procede al análisis de datos.

2.3.3 Visualización de datos en la plataforma de Kibana

Se disponen los datos en la plataforma concretamente (128 millones de datos) como se verá a continuación en la figura 25:

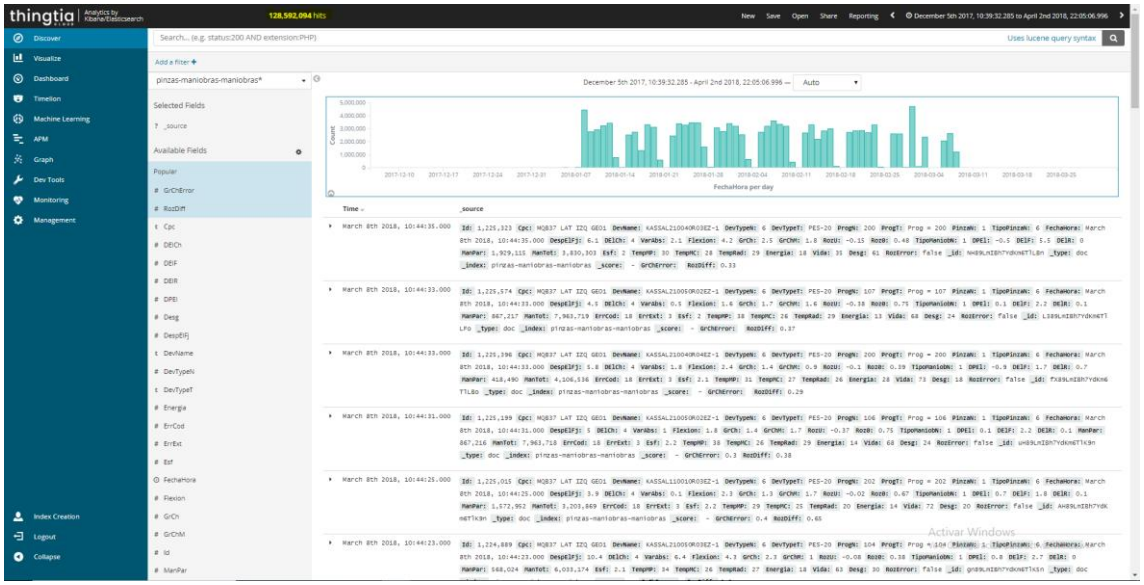


Figura: 25.Disposición datos Kibana.

No se tiene ninguna noción de que instalación ni que pinzas es la que más fallos tiene por rotura debido a valores fuera del rango soportado por la pinza, por tanto, una vez convertidos los Access en JSON y subidos a la plataforma se procede a realizar un **dashboard** (representación gráfica de las variables y formato requeridos por el usuario) como se observa en la figura 26 (se trata de una herramienta de Kibana capaz de hacer visualizaciones interactivas con la posibilidad de añadir filtros en una gran cantidad de datos).

En este caso el objetivo es encontrar aquella instalación que posee el robot con la pinza que en más ocasiones ha sobrepasado el rango soportado por la pinza [-0.4-0.4].

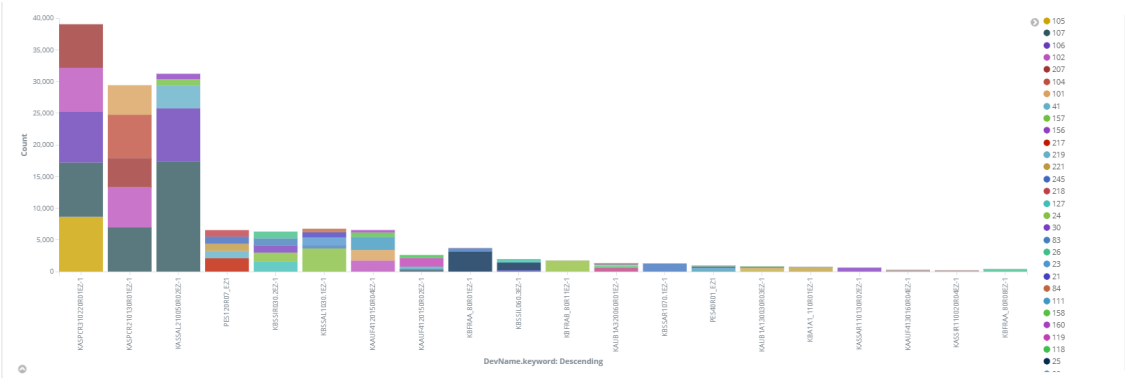


Figura: 26.Dashboard pinzas.

Como muestra la visualización del dashboard, la pinza más problemática (la pinza que más veces presenta un valor anómalo de rozamiento usado) es la "**KASPCR310220R01EZ-1**", y si se visualiza la instalación a la que pertenece esa pinza es a "**MQB37 MONTANTE C DCHO**". Esa pinza trabaja numerosos programas, todos ellos con la misma funcionalidad, pero diferentes trayectorias en función del lugar en el que se produce el fresado en la carrocería del automóvil. A continuación, se muestra de manera esquemática el lugar en el que está el montante C en la carrocería de un automóvil. Esa pinza se encarga de realizar los fresados en la parte interior del montante C. Por último, se muestra un esquema en la figura 27 del vehículo en el que se aprecian las diferentes partes de un vehículo en términos de fabricación y producción.

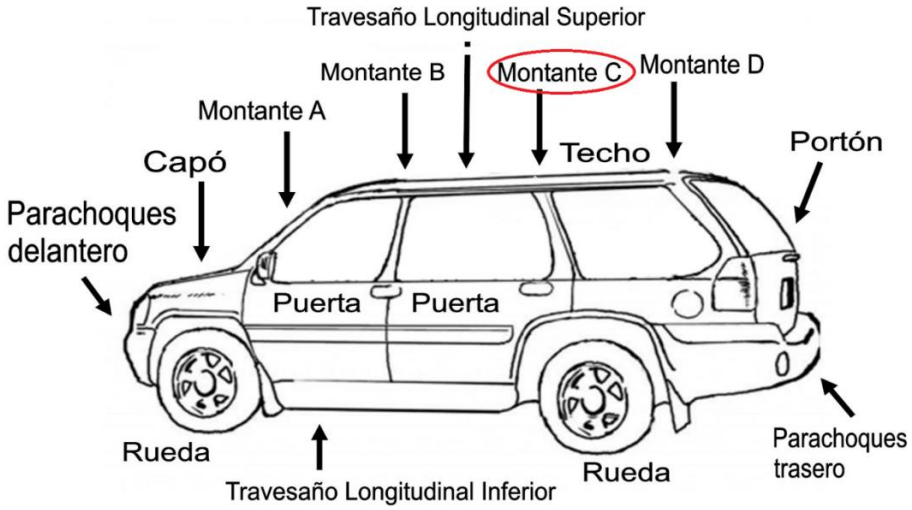


Figura: 27.Esquema automóvil.

Una vez detectada la pinza más problemática y su funcionalidad, se retoman aquellos campos que intervienen en el valor del rozamiento usado (RozU) de la pinza. Como se ha mencionado antes estos campos son: posición del electrodo fijo (**DespElFj**), posición del electrodo de chapa (**DElCh**), grosor de chapa medido (**GrChM**) y grosor de chapa programado (**GrCh**).

Se han realizado unos scripts en Kibana de manera que se unifiquen los 4 campos en solamente 2 para mayor capacidad de trabajo. En primer lugar, se procede a la creación de **VarAbs**, variable que resulta de la resta entre la posición del electrodo fijo (en valor absoluto) y la posición del electrodo de chapa (en valor absoluto), que en el caso de exceder de 3 indicaría una afectación del **RozU**. La otra variable creada es **GrChError,** resultado de la resta entre el grosor de chapa medido y programado. Este resultado que tiene que salir negativo en condiciones normales dado que el grosor de chapa programada siempre debe superar el medido. De este modo se consigue una reducción de la dimensionalidad del problema.

Estos 2 scripts se realizan en Kibana de manera que a través de todos los datos subidos va cogiendo forma las variables programadas mediante los scripts, como se aprecia en la figrua 28.

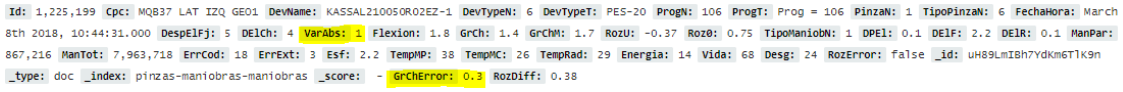


Figura: 28.Ejemplo de log pinza.

Se pueden observar las 2 variables introducidas, por tanto, una vez preparadas las variables y el foco del estudio localizado, se exportan los datos en formato Excel para tratarlos con la herramienta Matlab. Cada campo utilizado estará en un fichero por separado y en esa columna en la que se encuentra el valor solo habrá contenido

numérico dado que Matlab en caso de reconocer cualquier "*string*" (cadena) en la columna tomaría toda la columna como "*string*" (cadena) mientras que lo realmente interesante es el valor que ha dado la pinza numéricamente.

Capítulo 4

4 MÉTODO Y ALGORITMO EMPLEADOS

4.1 Introducción y elección algoritmo

En primer lugar, una vez se han revisado los algoritmos de aprendizaje automático, existe la necesidad de elegir aquel método que más se ajusta a las especificaciones del TFG.

En el caso de este estudio, nos encontramos con variables continuas de manera que la variable objetivo es un valor numérico continuo que determinarse en función de las entradas del modelo en lugar de limitarse a clasificarlo en posibles etiquetas. Una vez descartado todo algoritmo de clasificación, dados los continuos cambios en la instalación y la incorporación de nuevas tecnologías se hace necesario utilizar un algoritmo que presente una capacidad de respuesta rápida tanto para realizar el entrenamiento de los datos como para el conjunto de pruebas. Este hecho unido a que en este estudio la variable objetivo o de salida (RozU) depende de un número reducido de variables de entrada (VarAbs y GrChError), se traduce en que el modelo no debe tener una alta complejidad.

Además, el objetivo principal de este TFG es determinar el valor numérico del rozamiento en cada instante lo cual permita observar la tendencia del rozamiento en un ciclo de la pinza para poder sustituirla en el momento en el que la pinza muestre un deterioro excesivo. Por ello un ajuste correcto del algoritmo proporcionará una estimación bastante precisa respecto a los valores reales. Debido a todas estas razones se ha decidido desarrollar un algoritmo de regresión lineal basado en mínimos cuadrados por su gran robustez y por simplicidad para buscar una relación entre un número bajo de variables de entrada y una única variable de salida continua.

4.2 Regresión lineal

La regresión lineal es una técnica estadística que se utiliza para pronosticar el posible valor de una salida cuando su entrada toma un valor específico. La predicción la realiza aproximando, mediante el método de los mínimos cuadrados, una recta a los puntos que teníamos anteriormente recogidos que sirven de entrenamiento del modelo. Una vez obtenidos los parámetros del modelo, se realizan pronósticos del posible valor que puede tomar la salida dependiendo del valor que se quiera evaluar.

En este caso, el objetivo es predecir que rozamiento sufrirá la pinza, en un periodo concreto a analizar. Para ello se aplicará una regresión lineal en tres dimensiones, dado que ese rozamiento usado, dependerá tanto del grosor de chapa (GrCh) como de la variable absoluta (VarAbs). Anteriormente se han obtenido los resultados de distintas pinzas durante un funcionamiento normal de la cadena de producción de automóviles, estos valores son los que conocemos como conjunto de entrenamiento. También se puede observar el número de ensayos realizados (m) y el número de características (n) o número de datos de entrada que en este caso es 2 (***VarAbs y GrChError***). La salida en este caso es el rozamiento usado de la pinza alcanzado en cada momento.

Si se dispone de un conjunto de datos S = {(x^k , y^k), k = 1,…,N} , donde se tratan de predecir un valor numérico continuo, $y^k \in \mathbb{R}$, se puede emplear un modelo de regresión lineal. El objetivo que se persigue es encontrar la función h que mejor establezca la relación entre las variables independientes (descriptores) y la variable dependiente (etiqueta numérica). En el caso de la regresión lineal, la función h buscada se expresa como una combinación lineal de los descriptores con pesos θ y será de la forma:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m$$

El ajuste de los parámetros θ_1 ha de hacerse con el conjunto de datos disponibles S, de manera que $h_\theta(x^k)$ sea lo más próximo a y^k. Para calcular el error existente entre los datos reales y los datos estimados por nuestra función, se define una *función de coste*, el error cuadrático medio:

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (h_\theta(x) - y)^2$$

Donde N se corresponde con el número de datos. Posteriormente se deben escoger los parámetros θ que minimizan la función de coste $J(\theta)$. Para ello, en primer lugar, se comienza inicializando aleatoriamente los parámetros para después, emplear un algoritmo de búsqueda, como puede ser el algoritmo de gradiente de descenso, que sucesivamente cambie el valor de θ de forma que el valor de $J(\theta)$ sea cada vez menor y converja al valor de los parámetros que minimiza la función de coste.

El algoritmo de descenso por gradiente comienza con valor arbitrario para los parámetros, que se actualizan de la forma:

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta)$$

Donde α es la tasa de aprendizaje, que indica cuánto cambian los parámetros y el signo menos indica que se mueve en la dirección opuesta al gradiente (ya que la dirección del gradiente es la de máximo incremento de la función). Si se elige un valor de α muy pequeño, la función tardará mucho en converger. En cambio, si se escoge un valor muy

elevado puede oscilar y no llegar el converger el mínimo global esperado. En el caso de este estudio, el problema de regresión se resolverá por mínimos cuadrados aplicado en una función matricial con 2 parámetros.

A continuación, se muestra un ejemplo de la función de coste $J(\theta)$ donde se puede observar que es una función convexa con un mínimo global, como se puede apreciar en la figura 29:

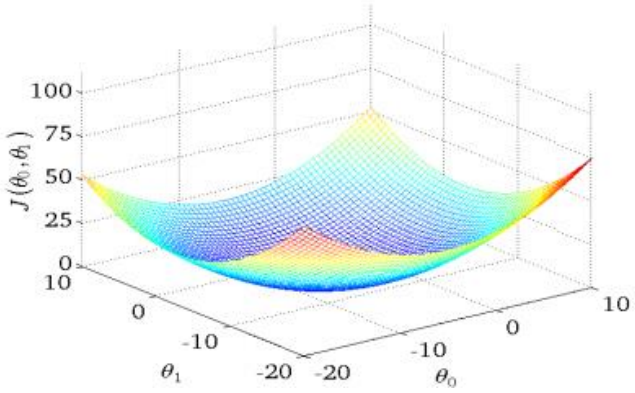


Figura: 29.Fn. de coste en regresión lineal.

4.2.1 Mínimos cuadrados (Least Squares)

Los mínimos cuadrados pretenden buscar la mejor función posible que relacione la variable dependiente con las independientes, es decir, la que mejor se aproxime a los datos con el criterio de mínimo error cuadrático.

Partiendo de la suposición de que los escalares $y_k, y_k = 0,1,..$ representan la salida (muestreada) de un sistema dinámico. Se puede suponer que aproximadamente el valor de la salida y_k puede ser calculado de la siguiente manera:

$$y_k, y_k \approx m_k\theta, k = 0,1, \ldots$$

Donde $m_k \epsilon \mathbb{R}^{1xnm}$ es un vector fila al que se le conoce como regresor, el regresor está formado por valores pasados de las entradas y salidas del sistema θ es un vector columna que contiene los coeficientes que relacionan a las salidas y las entradas. También se pueden considerar las perturbaciones medibles y las acciones de control como entradas.

Vamos a suponer un sistema con salida y_k, una entrada u; la relación entre las dos vendría dada por la siguiente ecuación

$$y_k = a_1 y_k - 1 + b_1 u_k - 1 + b_2 u_k - 2$$

El regresor para un instante k del tiempo de muestreo sería el siguiente:

$$m_k = [y_k - 1u_k - 1\ u_k - 2].$$

El vector paramétrico θ sería:

$$\theta = [a_1\ b_1 b_2]$$

Ahora se trata de calcular el vector de parámetros θ a partir de los valores de salida y_k y su regresor correspondiente m_k. Siempre no se va a poder obtener un regresor, en este ejemplo sería imposible construir el regresor m_0, ya que este depende de los valores pasados $y_{k-1}u_{k-1}$ y u_{k-2} valores imposibles de calcular. Por lo tanto, el primer regresor que podríamos calcular en este caso sería $m_2 = [y_1\ u_1 u_0]$. De esta manera, se puede asumir que el primer regresor será (yn, mn), donde n será en la mayoría de los casos mayor que 0.

Para el cálculo del error cuadrático en cada instante:

$$e_k = y_k - m_k\theta$$

Partiendo ahora de N pares (y(k), m(k)), se definen las siguientes matrices:

$$\mathrm{M(N)} = \begin{bmatrix} m(n) \\ \vdots \\ m(N) \end{bmatrix}$$

Sistema de ecuaciones:

$$E(N, \theta) = [e(n, \theta) \dots e(N, \theta)]^T$$

$$Y(N) = [y(n) \dots y(N)]^T$$

$$E(N, \theta) = Y(N) - M(N)\theta$$

La función para minimizar será la suma de todos los errores al cuadrado:

$$J(\theta) = ||[E(n, \theta)||^2 = \sum_{k=n}^{N} e^2(k, \theta)$$

$J(\theta)$ también puede definirse de manera matricial:

$$J(\theta) = (Y(N) - M(N)\theta)^T (Y(N) - M(N)\theta)$$

Para obtener el mínimo, hay que buscar el valor de θ que anule la derivada de la función de coste:

$$\frac{dJ(\theta)}{d\theta} = 0$$

Calculada la derivada e igualada a 0:

$$2(M(N)\theta - Y(N)M(N) = 0$$

El valor óptimo será, el llamado *estimador de mínimos cuadrados:*

$$\theta^* = [M^T(N)M(N)]^{-1}M^T(N)Y(N)$$

Para que se cumpla la ecuación anterior el producto $M(N)^T M(N)$ tiene que ser invertible. El mismo caso se da si el número de muestras no es suficiente, o si no se cumplen las condiciones de excitación persistente. La matriz puede ser no invertible o estar muy mal acondicionada (en la práctica se usa el PRBSS, una secuencia binaria que, si bien se genera con un algoritmo determinista, es difícil de predecir y exhibe un comportamiento estadístico similar a una secuencia verdaderamente aleatoria, es muy utilizado en telecomunicaciones, encriptación, simulación, técnica de correlación y espectroscopia en tiempo de vuelo).

A este método se le conoce como "**fuera de línea**", el cual se caracteriza por:

- Usualmente se necesitan muchas medidas para minimizar el efecto de ruidos y perturbaciones.
- Problemas de cálculo (inversión de una matriz).
- No es apropiado en sistemas cuya dinámica varía a lo largo del experimento.

4.2.2 Mínimos cuadrados recursivos (Recursive Least Squares)

El método anterior es un cálculo perfecto cuando ya se dispone de todos los datos (uso offline o "fuera de línea"). En cambio, su uso continuado en tiempo real (uso online o "en línea") es muy costoso en cuanto a tiempo de cálculo y almacenamiento de datos. Sin embargo, puede modificarse para obtener una estimación de los parámetros en cada instante de muestreo, actualizando con los nuevos valores disponibles, con pocos cálculos y sin tener que guardar más que unos pocos valores. Dicho algoritmo se conoce como Mínimos Cuadrados Recursivo (Recursive Least Squares, RLS)

Este método tiene menos carga de cálculo, usa las medidas que se van recogiendo, y es apropiado para sistemas cuya dinámica va variando.

La aplicación de mínimos cuadrados a sistemas variantes con el tiempo es de gran interés. Los parámetros que relacionan las salidas con las entradas también variaran con el tiempo. Se denomina θ_k la estimación del vector paramétrico en el tiempo de muestreo k.

En los mínimos cuadrados se obtiene θ_k para minimizar la función:

$$J_k(\theta) = \sum_{i=n}^{k} \lambda^{k-1}(y_i - m_i\theta)^2$$

Donde $\lambda \in (0; 1]$ se denomina factor de olvido. Si este valor fuera igual a 1, se obtendría lo mismo que en los mínimos cuadrados fuera de línea anteriormente explicados; afectando todos los errores de la misma manera. A medida que el valor λ va alejándose del 1, los valores alejados temporalmente se ponderan en menor medida, para obtener buenos resultados. Se definen las siguientes matrices:

$$Y_k = \begin{bmatrix} y_n \\ y_{n+1} \\ \vdots \\ y_k \end{bmatrix}, \; M_k = \begin{bmatrix} m_n \\ m_{n+1} \\ \vdots \\ m_k \end{bmatrix}, \; W_k = \begin{bmatrix} \lambda^{k-n} & & & \\ & \lambda^{k-n-1} & & \\ & & \ddots & \\ & & & \lambda \\ & & & & 1 \end{bmatrix}$$

La función a minimizar es la siguiente:

$$J_k(\theta) = (Y_k - M_k\theta)^T W_k (Y_k - M_k\theta)$$

Para obtener el valor óptimo de θ_k se analiza la función para una perturbación

$$\theta k + \Delta\theta$$

respecto del óptimo:

$$\begin{aligned} J_k(\theta_k + \Delta\theta) &= (Y_k - M_k(\theta_k + \Delta\theta))^T W_k\big(Y_k - M_k(\theta_k + \Delta\theta)\big) \\ &= (Y_k - M_k\theta_k)^T W_k (Y_k - M_k\theta_k) + \Delta^T\theta {M_k}^T W_k M_k \Delta\theta_k - (Y_k \\ &- M_k\theta_k)^T W_k M_k \Delta\theta - \Delta\theta^T {M_k}^T W_k (Y_k - M_k\theta_k) \\ &= J_k(\theta_k) + \Delta\theta^T {M_k}^T W_k M_k \Delta\theta_k - 2\Delta\theta^T {M_k}^T W_k (Y_k - M_k\theta_k) \end{aligned}$$

Eligiendo un valor de θ tal que todos los términos $\Delta\theta_k$ se eliminen (esto es equivalente a hacer cero el gradiente respecto de θ_k):

$${M_k}^T W_k (Y_k - M_k\theta_k) = 0$$

De aquí se extrae que

$${M_k}^T W_k Y_k = {M_k}^T W_k M_k \theta_k$$

O equivalentemente:

$$\theta_k = ({M_k}^T W_k M_k)^{-1} {M_k}^T W_k Y_k$$

Por lo que la ecuación del coste para $\theta_k + \Delta\theta$ es:

$$J_k(\theta_k + \Delta\theta) \ = J_k(\theta_k) + \ \Delta\theta {M_k}^T W_k M_k \Delta\theta_k \geq J_k(\theta_k), \forall \ \Delta\theta$$

Garantizando que

$$\theta_k = ({M_k}^T W_k M_k)^{-1} {M_k}^T W_k Y_k$$

Es óptima en el sentido de los mínimos cuadrados ponderados.

Ahora se va a presentar una forma recursiva con la que se pueden obtener valores de θ_k en cada instante k en función de θ_{k-1}. De esta manera la siguiente matriz:

$$P_k = \left(\sum_{i=n}^{k} \lambda^{k-1} {m_i}^T m_i \right)^{-1}$$

P_k es una matriz simétrica ya que está definida como la inversa de una matriz simétrica. Para obtener P_k de manera recursiva utilizaremos P_{k-1}:

$$P_{k-1} = \left(\sum_{i=n}^{k-1} \lambda^{k-1} {m_i}^T m_i \right)^{-1}$$

Ahora se obtiene la relación entre P_k y P_{k-1}:

$$\begin{aligned} {P_k}^{-1} &= \textstyle\sum_{i=n}^{k} \lambda^{k-1} {m_i}^T m_i \\ &= {m_k}^T m_k + \textstyle\sum_{i=n}^{k-1} \lambda^{k-1} {m_i}^T m_i \\ &= {m_k}^T m_k + \lambda \textstyle\sum_{i=n}^{k-1} \lambda^{k-1-i} {m_i}^T m_i \\ &= {m_k}^T m_k + \lambda {P^{-1}}_{k-1} \end{aligned}$$

Aplicando el lema de inversión se permite reescribir la relación entre P_k y P_{k-1} sin realizar la inversa. La ecuación queda de la siguiente manera:

$$P_k = \frac{1}{\lambda} \left(P_{k-1} - \frac{(m_k P_{k-1})^T (m_k P_{k-1})}{\lambda + m_k P_{k-1} {m_k}^T} \right)$$

Se puede demostrar que P_{k-1} es igual a la matriz ${M_k}^T W_k M_k$:

$$
\begin{aligned}
M_k^\top W_k M_k &= \begin{bmatrix} m_n^\top & \dots & m_{k-1}^\top & m_k^\top \end{bmatrix} \begin{bmatrix} \lambda^{k-n} & & & \\ & \ddots & & \\ & & \lambda & \\ & & & 1 \end{bmatrix} \begin{bmatrix} m_n \\ \vdots \\ m_{k-1} \\ m_k \end{bmatrix} \\
&= \begin{bmatrix} m_n^\top & \dots & m_{k-1}^\top & m_k^\top \end{bmatrix} \begin{bmatrix} \lambda^{k-n} m_n \\ \vdots \\ \lambda m_{k-1} \\ m_k \end{bmatrix} \\
&= \sum_{i=n}^{k} \lambda^{k-i} m_i^\top m_i = P_k^{-1}.
\end{aligned}
$$

Sustituyendo en la ecuación obtenemos:

El valor de θ_{k-1} será el siguiente:

$$
\begin{aligned}
\theta_k &= (M_k^\top W_k M_k)^{-1} M_k^\top W_k Y_k \\
&= P_k M_k^\top W_k Y_k \\
&= P_k \begin{bmatrix} m_n^\top & \dots & m_{k-1}^\top & m_k^\top \end{bmatrix} \begin{bmatrix} \lambda^{k-n} & & & \\ & \ddots & & \\ & & \lambda & \\ & & & 1 \end{bmatrix} \begin{bmatrix} y_n \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} \\
&= P_k \left(\sum_{i=n}^{k} \lambda^{k-i} m_i^\top y_i \right).
\end{aligned}
$$

Y el de $\theta_k$será:

$$
\begin{aligned}
\theta_k &= P_k \left(\sum_{i=n}^{k} \lambda^{k-i} m_i^\top y_i \right) \\
&= P_k \left(m_k^\top y_k + \sum_{i=n}^{k-1} \lambda^{k-i} m_i^\top y_i \right)
\end{aligned}
$$

Sustituyendo con la ecuación obtenemos θ_k en función de θ_{k-1}:

$$
\begin{aligned}
\theta_k &= \frac{1}{\lambda}\left(P_{k-1} - \frac{(m_k P_{k-1})^\top (m_k P_{k-1})}{\lambda + m_k P_{k-1} m_k^\top}\right) m_k^\top y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{1}{\lambda}\left(P_{k-1} m_k^\top - \frac{P_{k-1} m_k^\top (m_k P_{k-1} m_k^\top)}{\lambda + m_k P_{k-1} m_k^\top}\right) y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{1}{\lambda} P_{k-1} m_k^\top \left(1 - \frac{m_k P_{k-1} m_k^\top}{\lambda + m_k P_{k-1} m_k^\top}\right) y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{P_{k-1} m_k^\top y_k}{\lambda + m_k P_{k-1} m_k^\top} + \theta_{k-1} - \frac{P_{k-1} m_k^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \theta_{k-1} + \frac{P_{k-1} m_k^\top (y_k - m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top}.
\end{aligned}
$$

Finalmente se define la ganancia de estimación como:

$$K_k = \frac{P_{k-1} {m_k}^T}{\lambda + m_k P_{k-1} {m_k}^T}$$

Con la ganancia de estimación sustituimos en la matriz de covarianzas (P_k) y Θ_k. Obtenemos las siguientes ecuaciones para el cálculo recursivo de θ_k:

$$K_k = \frac{P_{k-1} {m_k}^T}{\lambda + m_k P_{k-1} {m_k}^T}$$

Actualización de la estimación de parámetros:

$$\theta_k = \theta_{k-1} + K_k (y_k - M_k \theta_{k-1})$$

Actualización de la matriz de covarianzas, se espera que exista una matriz de covarianza que debe evolucionar conforme aumenta la cantidad de mediciones de manera tal que el error en la estimación pueda ser minimizado cuando el conjunto de muestras sea lo suficientemente grande:

$$P_k = \frac{1}{\lambda}(1 - K_k m_k) P_{k-1}$$

Un λ próximo a 1 “recuerda” muchas medidas por lo que es más insensible al ruido, pero se adapta más lentamente. Un λ menor “recuerda” menos por lo que se adapta rápidamente, pero es más sensible al ruido.

Hay que tener una serie de consideraciones prácticas:

- Usando factor de olvido, si el punto de trabajo no varía P_k puede crecer demasiado.
- Solución a estos problemas: factor de olvido variable:
  - ✓ Si la traza de P supera el límite establecer λ igual a 1.
  - ✓ En caso de que P esté por debajo de un mínimo: disminuir λ o sumar una matriz positiva.
- Problemas numéricos pueden producir que P sea definida negativa.
- La solución pasa por factorizar P_k como:

$$P_k = U_k D_k P_k^T$$

donde D_k es diagonal y U_k es triangular superior con $U_{iik} = 1$.

4.3 Resultados

4.3.1 Mínimos cuadrados

Se ha programado un conjunto de datos en diferentes ventanas de longitudes temporales para comprobar la confianza de las medidas tanto a corto como a largo plazo:

- **Caso 1:**
  Conjunto de entrenamiento: *25 de enero – 2 de febrero (7 días)*
  Conjunto de prueba: *14 de febrero – 22 de febrero (7 días)*
  Las gráficas pertenecientes a este caso están representadas a continuación. En la figura 30 se muestra la predicción y en la figura 31 los errores existentes.

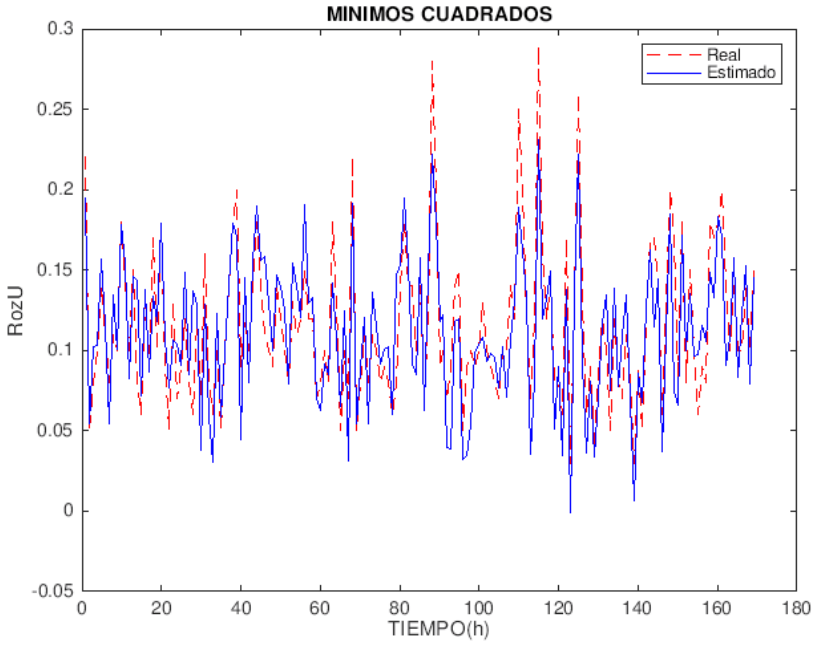


Figura: 30.Caso 1. Predicción.

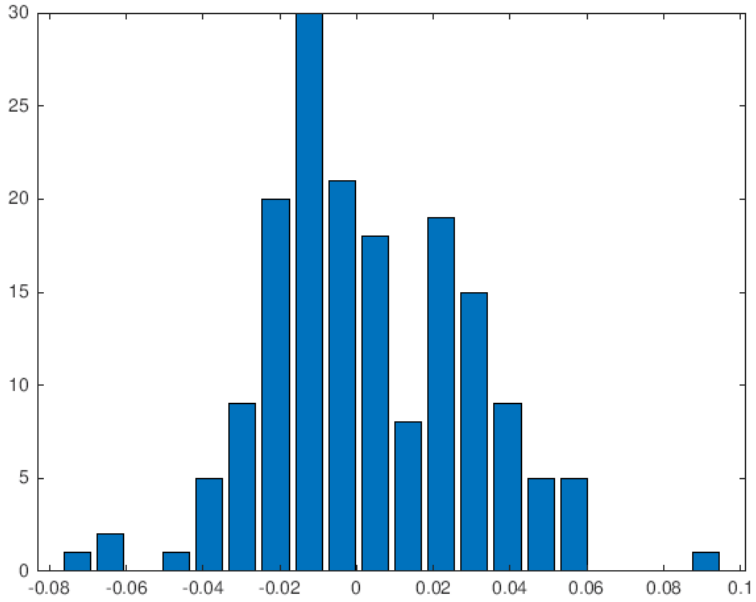


Figura: 31.Errores Caso 1.

Tabla de errores:

	MAPE (%)	RMSE	MAE
Caso 1	13.21	0.0061	0.0212

Tabla 1.Errores Caso 1.

- **Caso 2:**

  Conjunto de entrenamiento: *24 de enero – 6 de febrero (10 días)*

  Conjunto de prueba: *14 de febrero – 27 de febrero (10 días)*

  Las gráficas pertenecientes a este caso están representadas a continuación. En la figura 32 se muestra la predicción y en la figura 33 los errores existentes.

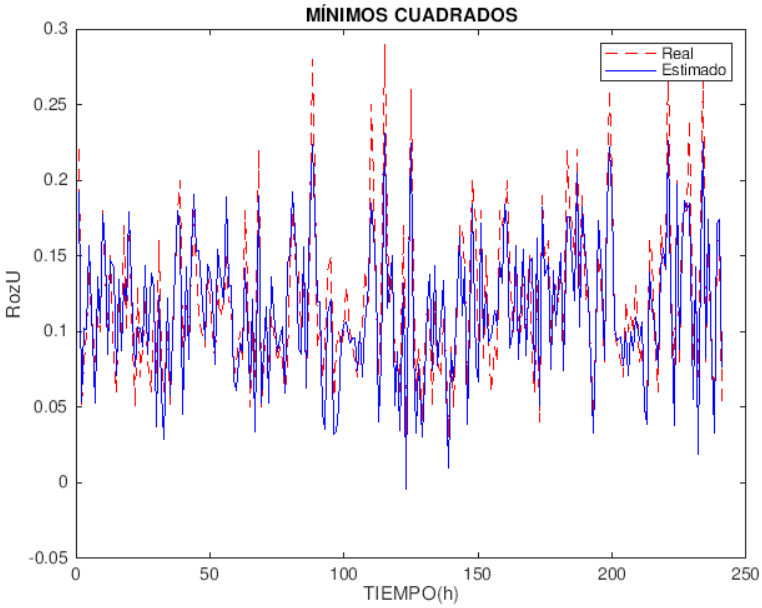


Figura: 32.Caso 2. Predicción.

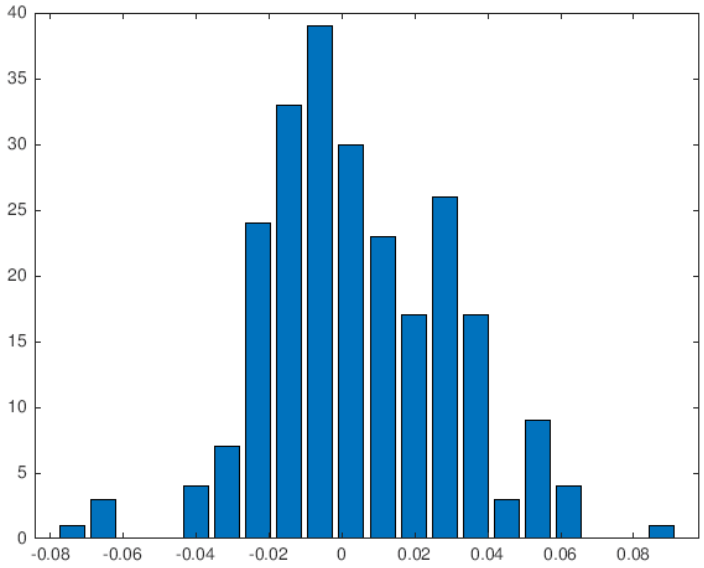


Figura: 33.Errores Caso 2.

Tabla de errores:

	MAPE (%)	RMSE	MAE
Caso 2	15.21	0.0046	0.0203

Tabla 2. Errores Caso 2.

- **Caso 3:**

Conjunto de entrenamiento: *17 de enero – 6 de febrero (15 días)*
Conjunto de prueba: *7 de febrero – 27 de febrero (15 días)*
Las gráficas pertenecientes a este caso están representadas a continuación. En la figura 34 se muestra la predicción y en la figura 35 los errores existentes.

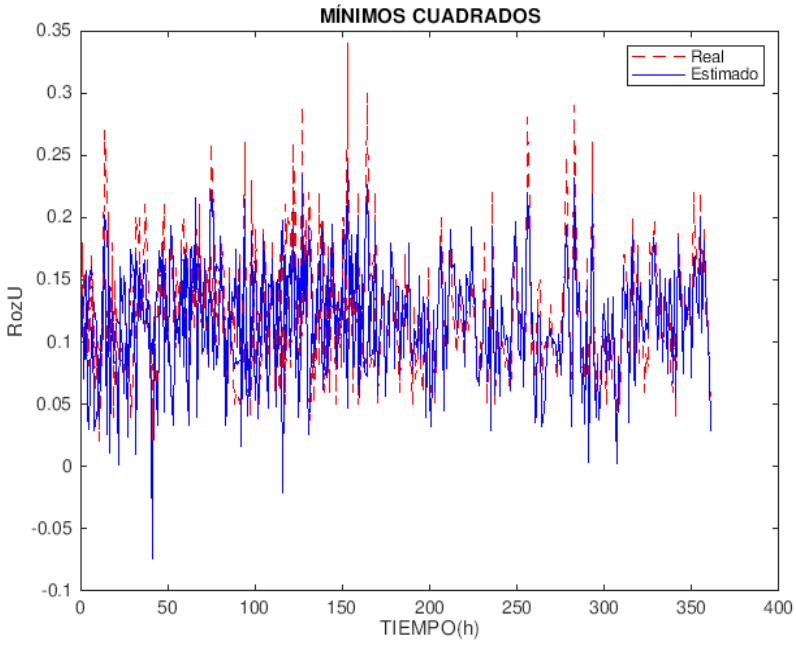


Figura: 34.Caso 3. Predicción.

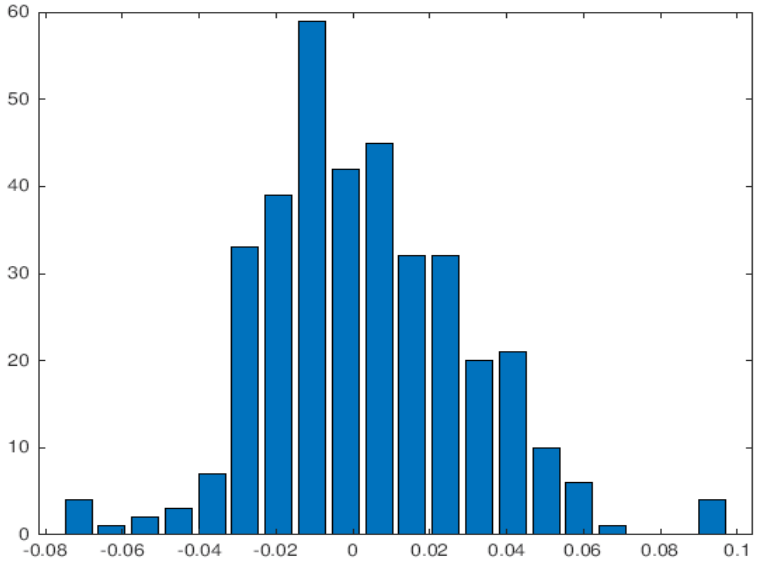


Figura: 35.Errores Caso 3.

Tabla de errores:

	MAPE (%)	RMSE	MAE
Caso 3	14.26	0.0029	0.0211

Tabla 3. Errores Caso 3.

4.3.1.1 Conclusiones mínimas cuadrados

A continuación, se analizan los errores anotados en las tablas 1,2 y 3, en relación con las estimaciones del análisis realizado.

El **error porcentual absoluto medio (MAPE)** es muy utilizado a la hora de realizar pronósticos debido a su fácil interpretación. En este caso está en torno al 15%, en los 3 casos, es ligeramente alto dado que se están estimando valores de hora en hora para darle mayor visibilidad, como veremos más adelante este valor bajará en el caso de un ciclo completo, analizado minuto a minuto.

El **error medio absoluto (MAE)** mide la magnitud promedio de los errores en un conjunto de pronósticos, sin considerar su tendencia. Mide la precisión para variables continuas. La ecuación se da en las referencias da los errores en el capítulo 2. Expresado de otra forma, el MAE es el promedio sobre el conjunto de verificación de los valores absolutos de las diferencias entre el pronóstico y la observación correspondiente. El MAE es un puntaje lineal, lo que significa que todas las diferencias individuales se

ponderan por igual en el promedio. En el caso del TFG como la variable de salida, el RozU oscila entre 0 y 1, el error medio absoluto tiene un valor muy bajo, en torno a 0.02 y por tanto una alta precisión en los pronósticos.

El **error cuadrático medio (RMSE)** es una regla de puntuación cuadrática que mide la magnitud promedio del error. La ecuación. En resumen, se trata de analizar la diferencia entre el pronóstico y los valores observados correspondientes, elevados al cuadrado para posteriormente promediarlos sobre la muestra y finalmente realizar la raíz cuadrado sobre este término.

Dado que los errores observados se elevan al cuadrado antes de promediarlos, el RMSE otorga un peso relativamente alto a los errores más elevados. Esto significa que el RMSE es más útil cuando se quieren desechar los errores y valores anómalos. En este caso esos errores más elevados pueden darse por ejemplo en el caso 3 cerca de las 50 horas de entrenamiento, el algoritmo estima un valor mucho menor del real. A ese tipo de errores les asigna más peso el RMSE.

El MAE y el RMSE se pueden usar juntos para diagnosticar la variación en los errores en un conjunto de pronósticos. A mayor diferencia entre ellos, mayor es la varianza en los errores individuales en la muestra. Si el RMSE = MAE, por tanto todos los errores son de la misma magnitud.

Tanto el MAE como el RMSE pueden tener un rango de 0 a ∞. Como era de esperar se han obtenido valores pequeños en el RMSE. Es un caso muy similar al MAE.

Por tanto, se tienen errores menores dando por aceptable el modelo estimado frente al real durante el entrenamiento. La lógica dice que, a mayor volumen de datos, más sencillo es “aprender” el conjunto de datos para poder estimar unos valores más adecuados, pero también cabe indicar que estamos ante un sistema real cuyo comportamiento no conocemos excesivamente y por tanto dado el corto espacio de tiempo del que se poseen datos, se han intentado estimar en los 3 casos el mismo periodo de tiempo para observar cuál de ellos es más fiable. Como podemos ver han causado menos error el menor y el mayor volumen de datos, siendo el caso 2, el intermedio, el que ha dado mayor error. Pudo ocurrir que durante ese periodo en el cual se estaban extrayendo los datos hubiera alguna anomalía y que en el siguiente ciclo durante el cual se probaron las estimaciones ya se corrigiera, estimando un valor que no era real debido a aquella anomalía. Por ello cuanto mayor sea el conjunto de datos y cuanto mayor número de ciclos de pinzas se cojan durante el TFG se podrá ir mejorando hasta obtener una gran exactitud en las predicciones.

La finalidad de la estimación de los valores del rozamiento de las pinzas es realizar un mantenimiento predictivo en lugar de un mantenimiento preventivo, es decir, detectar el momento en el que la pinza está llegando al fin de su vida útil para adelantarse a su

rotura y evitar pérdidas de tiempo durante el periodo de fabricación del coche. Por tanto, se han estudiado 3 casos (7 días, 10 días y 15 días) de hora en hora simplemente para comprobar que las estimaciones son lo más aproximadas posibles a la realidad y ver que el modelo y algoritmo generado son bastante aproximados de manera nítida y sin gran cantidad de datos alrededor.

Cabe reseñar que esta instalación automovilística inicia sus paradas cada fin de semana, y cada día festivo a nivel nacional (España), esa es la razón por la cual cuando se establece tanto el periodo de entrenamiento como el periodo de prueba los días establecidos son días hábiles, es decir, días en los que la cadena de producción está en funcionamiento, en lugar de días naturales. Esa es la razón también por la que el código introducido en Matlab cuando establecemos el mes tanto de prueba como de entrenamiento, el número de días que aparecen son aquellos en los que la cadena de producción está activa de ahí que oscilen entre los 20-22 días.

En el caso de mostrar un ciclo completo de la pinza, como se va a mostrar a continuación, donde los datos obtenidos por el robot son minuto a minuto, se hace muy complicado apreciarlo con exactitud, pero si la tendencia de la vida útil de la pinza y se puede observar claramente como poco a poco se va desajustando y deteriorándose con el uso hasta alcanzar valores de rozamiento tan elevados que provocan su rotura y por tanto se hace necesario su sustitución.

En primer lugar, se muestra el final del ciclo de una pinza mostrado en la figura 36 hasta su rotura, posteriormente se observa el funcionamiento normal de la pinza tras ser reemplazada, hasta su posterior deterioro, y por último rotura. Las últimas trazas que se pueden observar en la gráfica ya son tras haber sido reemplaza de nuevo.

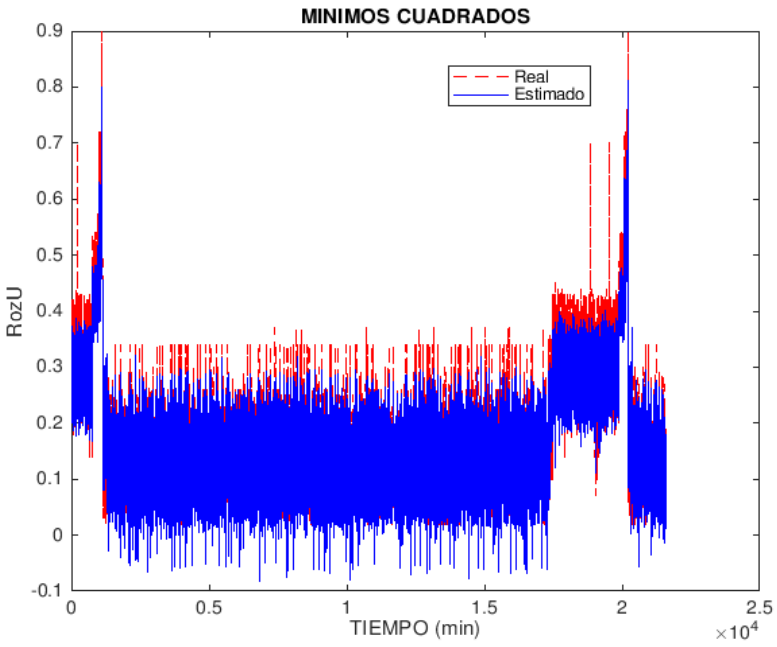


Figura: 36.Ciclo pinza. Predicción.

Como podemos ver a continuación en la figura 37, la mayor cantidad de error se concentra en torno al valor 0, siendo este el error más encontrado. El tiempo analizado en este caso es en torno a 2,5 x10^4 min, que equivale a aproximadamente, 18 días.

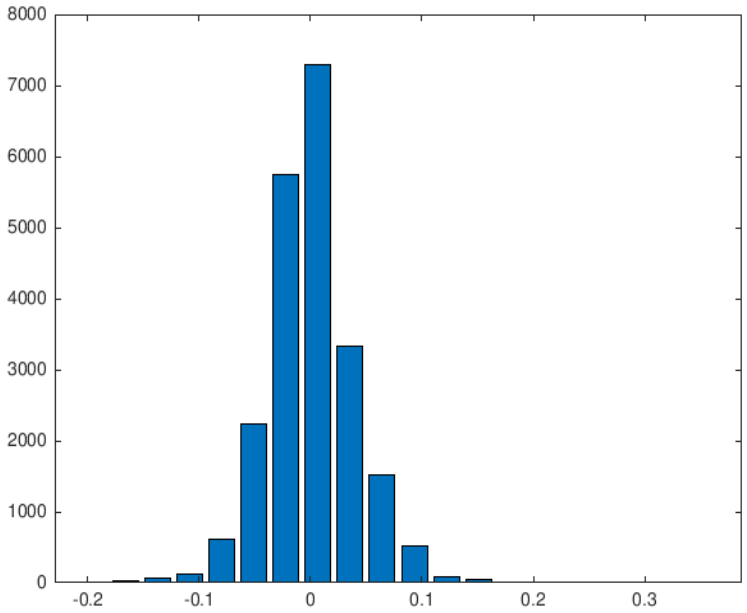


Figura: 37.Errores Ciclo pinza.

4.3.2 Mínimos cuadrados recursivos

Se han realizado todos los casos con tres factores de olvido diferentes: λ=0.95 λ=0,90 y λ=0.85.

Muchas aplicaciones pueden considerarse como sistemas de parámetros variables. En ellos se requiere que los parámetros de modelo estimado se vayan adaptando en función de los cambios del sistema. Interesa en estos casos controlar el cómo las medidas antiguas o las recientes afectan en la estimación de los parámetros o la rapidez con las que son olvidadas las medidas anteriores o antiguas.

Mientras más se aleje el valor de λ de 1, los valores más cercanos a la muestra cobraran más importancia que los demás, será más sensible al ruido, pero por el contrario se adaptará más rápidamente.

Resultados:

- **Caso 1:**

  Para λ=0.95

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 38 se muestra la predicción y en la figura 39 los errores existentes.

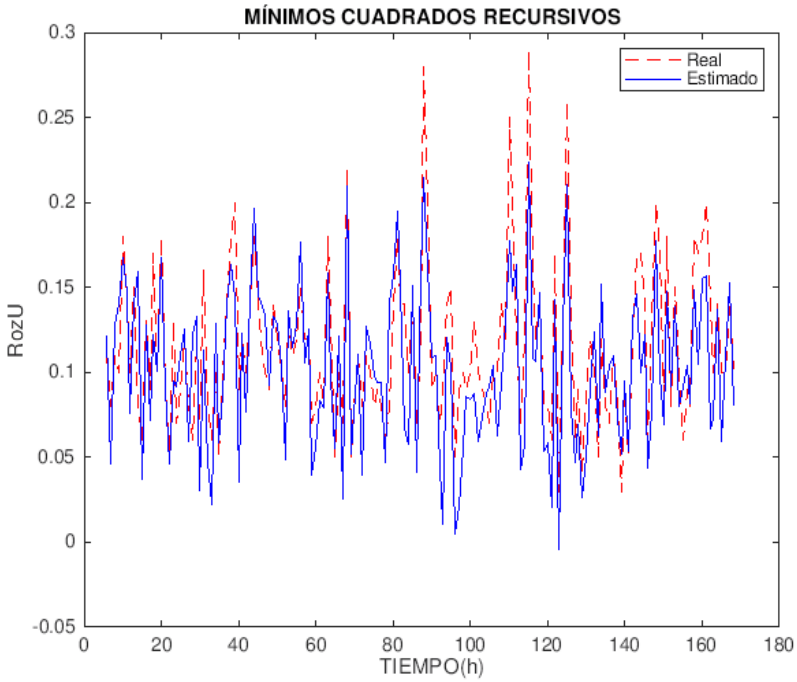


Figura: 38.Caso 1 mínimos cuadrados recursivos λ=0.95.

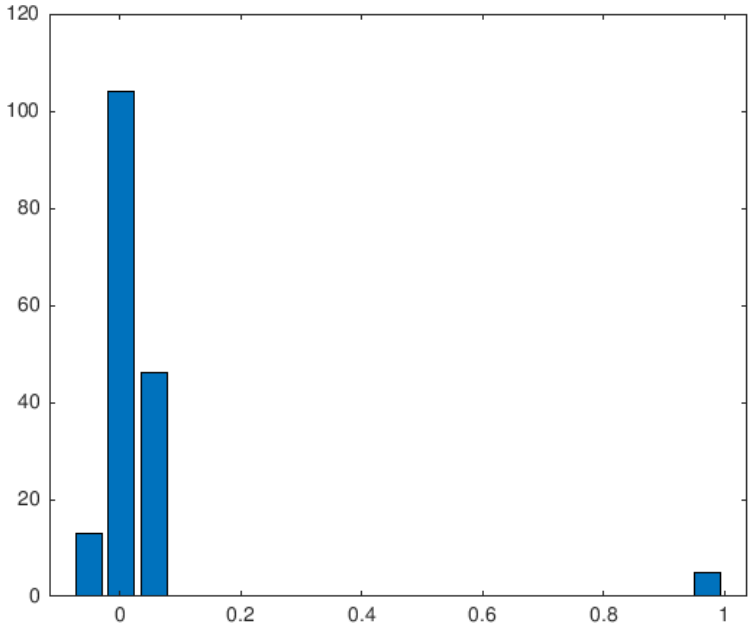


Figura: 39.Errores Caso 1 mínimos cuadrados recursivos λ=0.95.

- **Caso 1:**

  Para λ=0.90

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 40 se muestra la predicción y en la figura 41 los errores existentes.

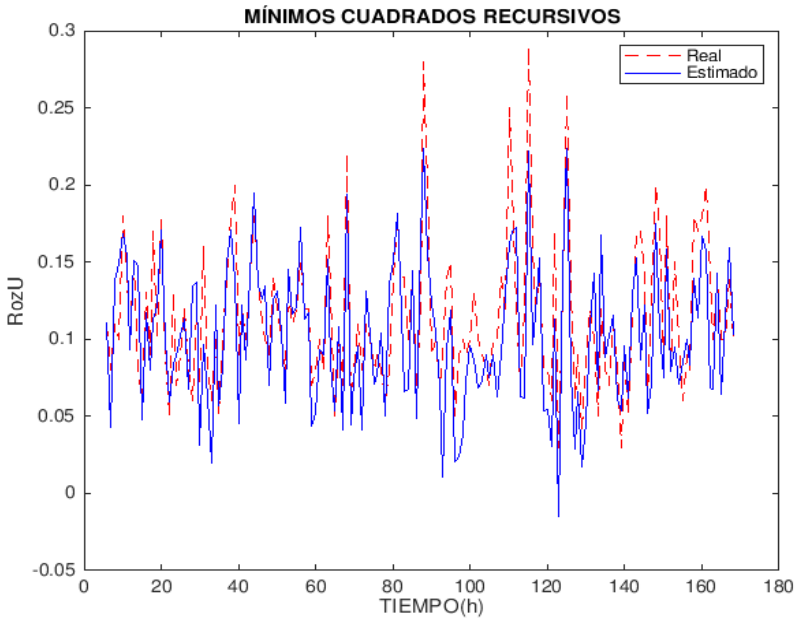


Figura: 40. Caso 1 mínimos cuadrados recursivos λ=0.90.

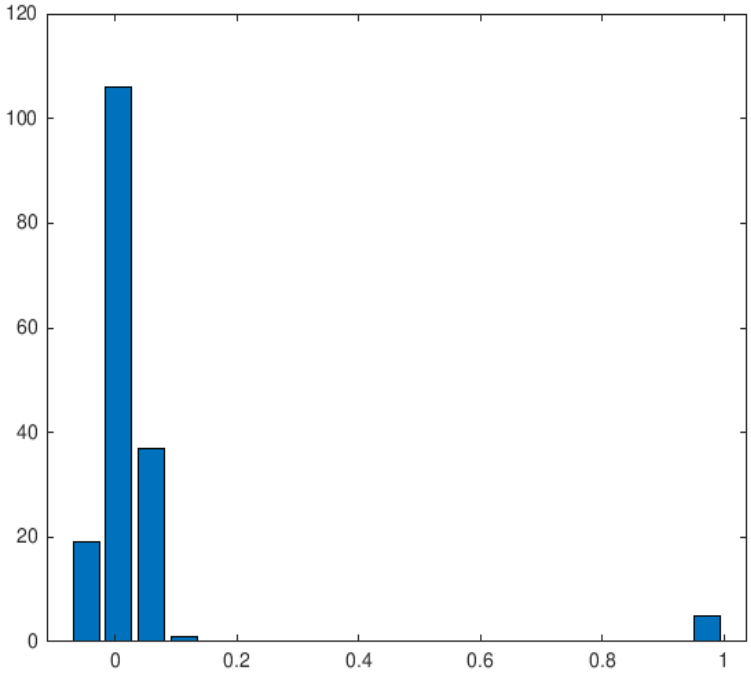


Figura: 41. Errores Caso 1 mínimos cuadrados recursivos λ=0.90.

- **Caso 1:**

  Para λ=0.85

  Las gráficas pertenecientes a este caso están representadas a continuación. En la figura 42 se muestra la predicción y en la figura 43 los errores existentes.

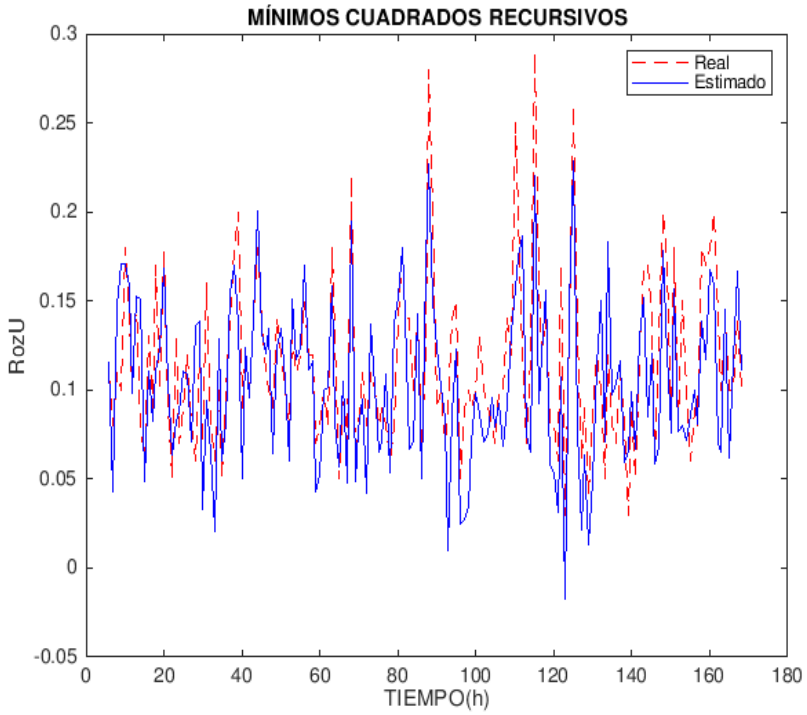


Figura: 42. Caso 1 mínimos cuadrados recursivos λ=0.85.

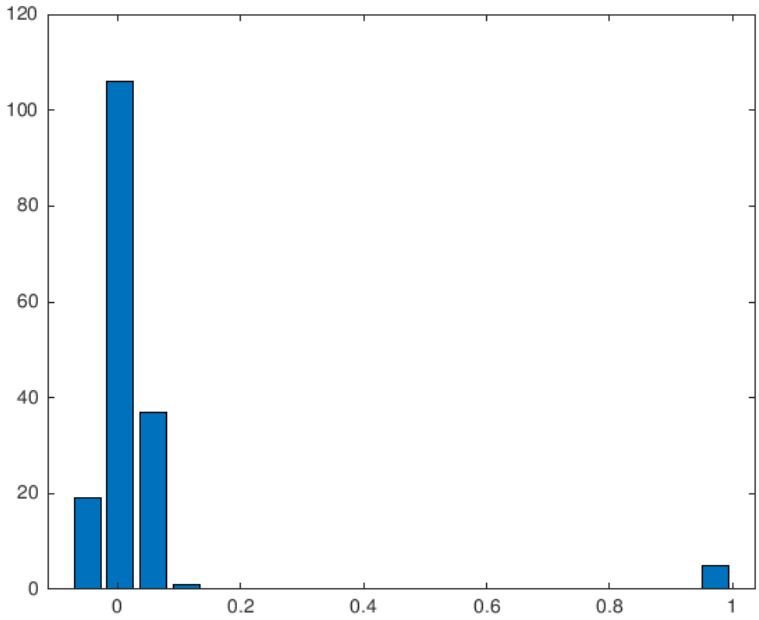


Figura: 43. Errores Caso 1 mínimos cuadrados recursivos λ=0.85.

	MAPE (%)	RMSE	MAE
Caso 1	13.21	0.0061	0.0212
Caso 1 (λ=0.95)	15.22	0.0063	0.0245
Caso 1 (λ=0.90)	17.23	0.0062	0.0247
Caso 1 (λ=0.85)	15.20	0.0063	0.260

Tabla 4. Caso 1 Errores recursivos.

- **Caso 2:**

  Para λ=0.95

  Las gráficas pertenecientes a este caso están representadas a continuación. En la figura 44 se muestra la predicción y en la figura 45 los errores existentes.

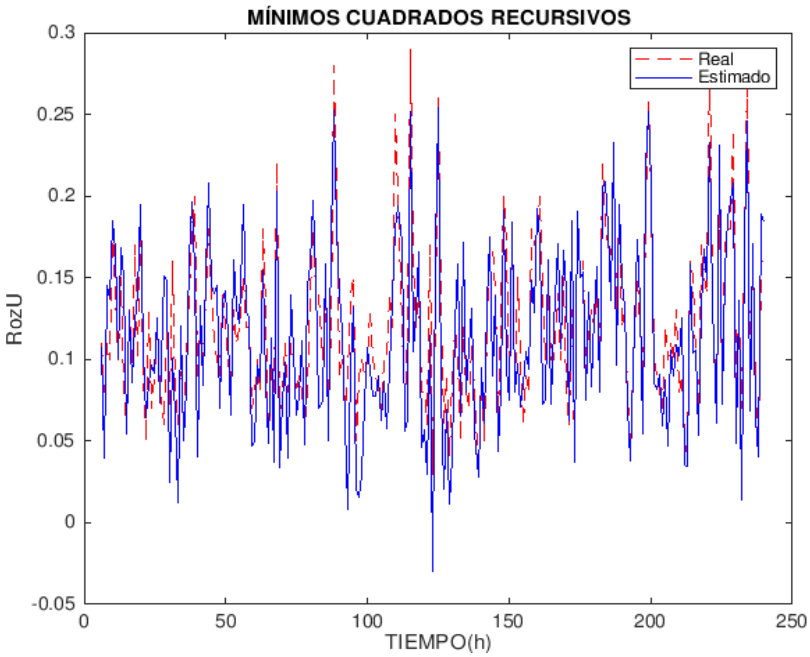


Figura: 44. Caso 2 mínimos cuadrados recursivos λ=0.95.

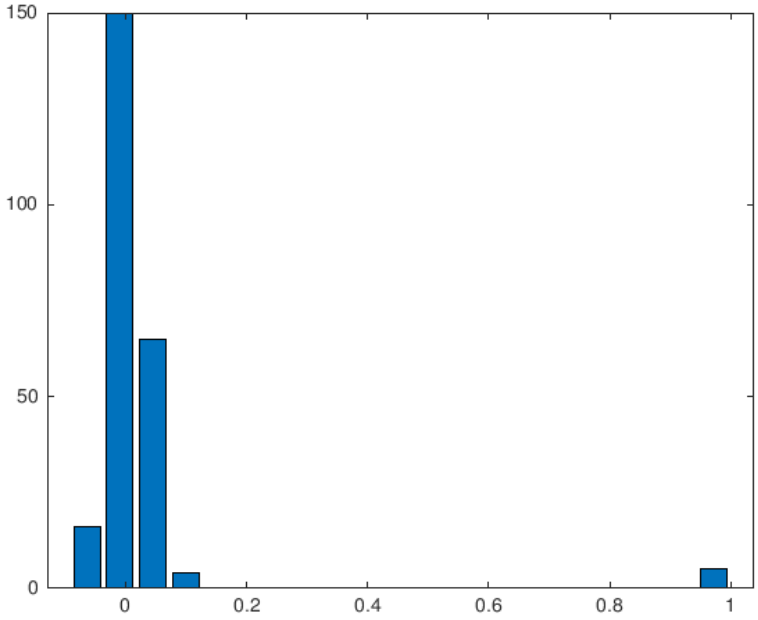


Figura: 45. Errores Caso 2 mínimos cuadrados recursivos λ=0.95.

- **Caso 2:**

  Para λ=0.90

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 46 se muestra la predicción y en la figura 47 los errores existentes.

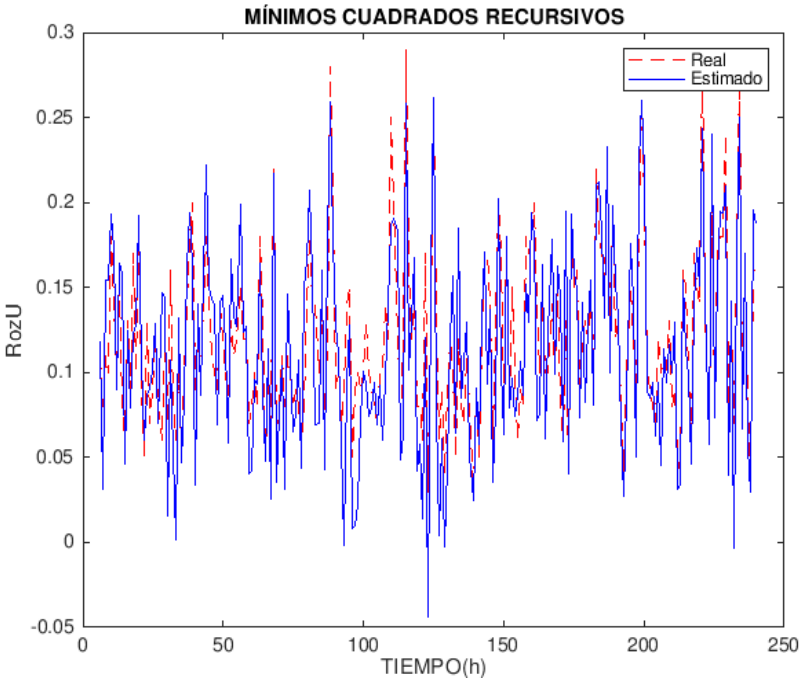


Figura: 46. Caso 2 mínimos cuadrados recursivos λ=0.90.

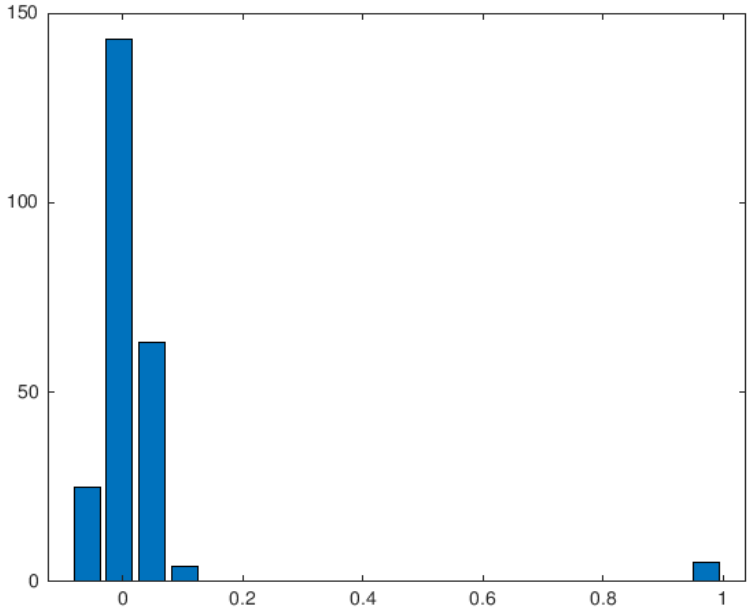


Figura: 47. Errores Caso 2 mínimos cuadrados recursivos λ=0.90.

- **Caso 2:**

  Para λ=0.85

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 48 se muestra la predicción y en la figura 49 los errores existentes.

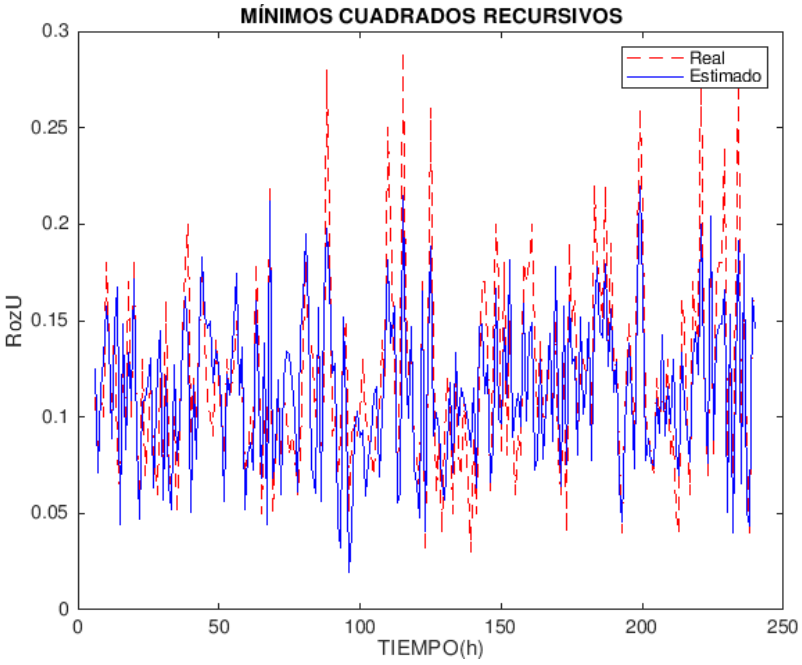


Figura: 48. Caso 2 mínimos cuadrados recursivos λ=0.85.

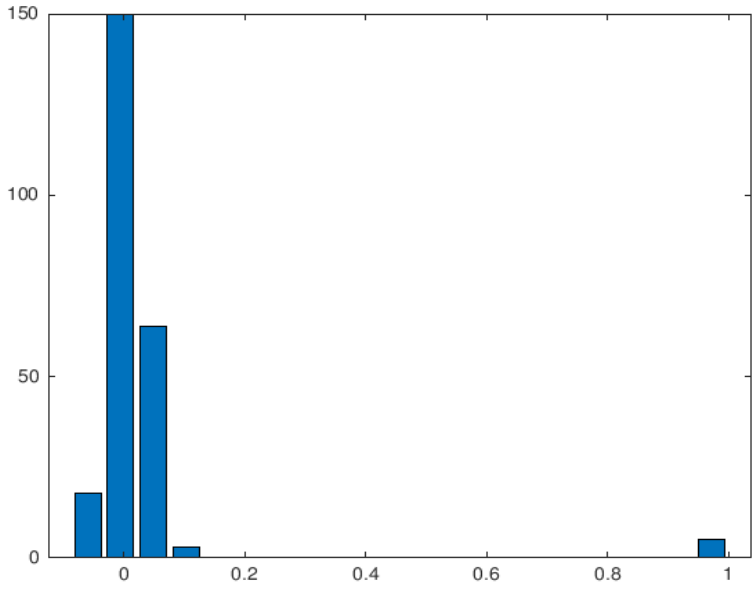


Figura: 49. Errores Caso 2 mínimos cuadrados recursivos λ=0.85.

	MAPE (%)	RMSE	MAE
Caso 2	15.21	0.0046	0.0203
Caso 2 (λ=0.95)	16.23	0.0047	0.0234
Caso 2 (λ=0.90)	17.26	0.0048	0.0251
Caso 2 (λ=0.85)	16.35	0.044	0.0235

Tabla 5.Caso 2 Errores recursivos.

- **Caso 3:**

Para λ=0.95

Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 50 se muestra la predicción y en la figura 51 los errores existentes.

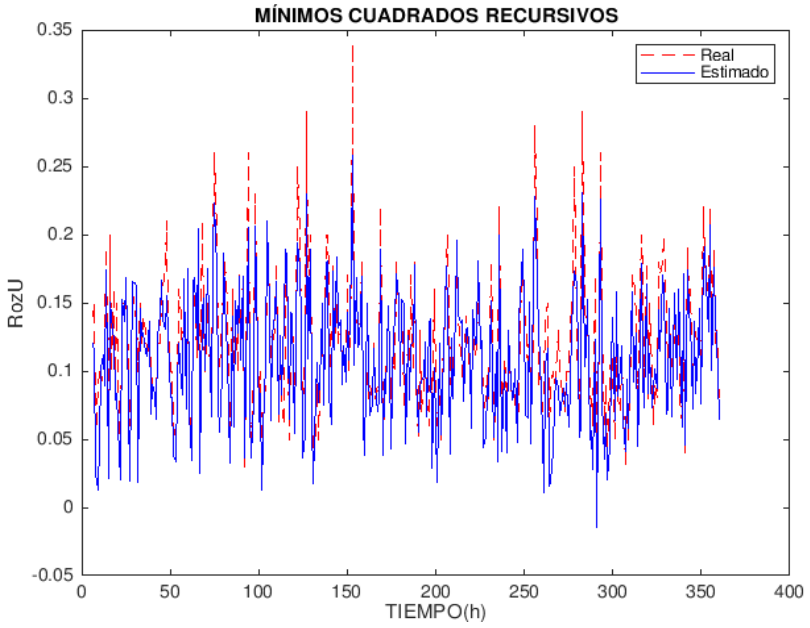


Figura: 50. Caso 3 mínimos cuadrados recursivos λ=0.95.

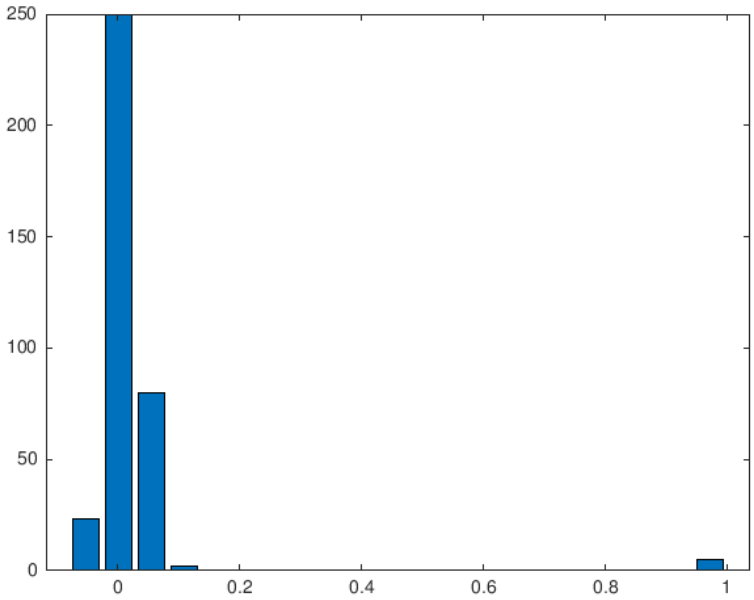


Figura: 51. Errores Caso 3 mínimos cuadrados recursivos λ=0.95.

- **Caso 3:**

  Para λ=0.90

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 52 se muestra la predicción y en la figura 53 los errores existentes.

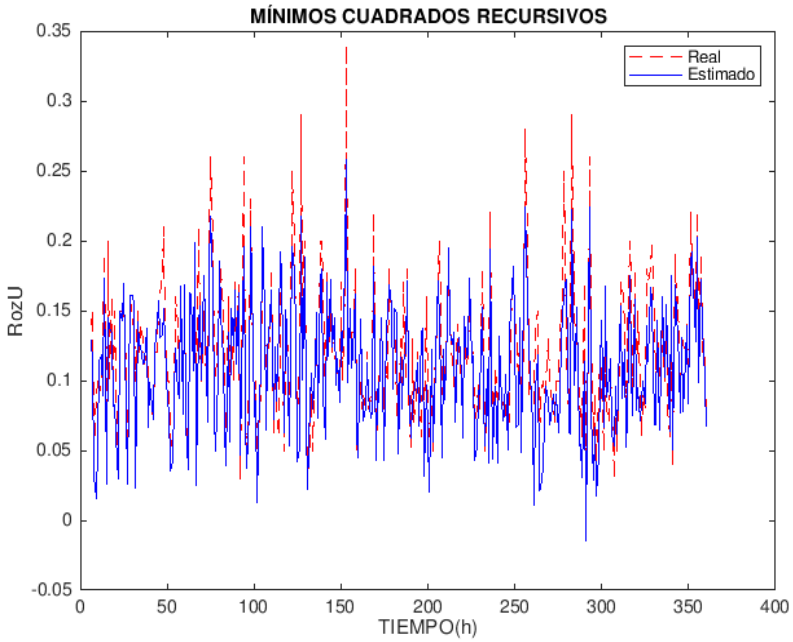


Figura: 52.. Caso 3 mínimos cuadrados recursivos λ=0.90.

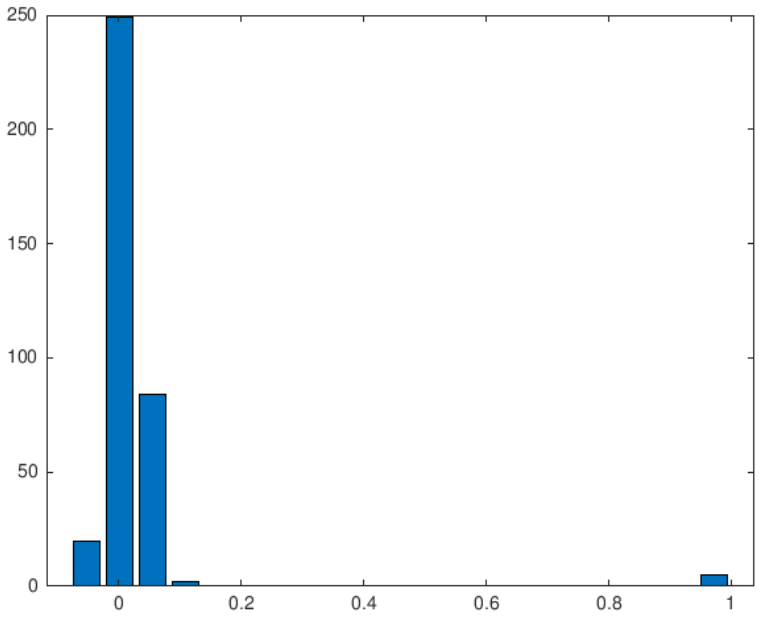


Figura: 53. Errores Caso 3 mínimos cuadrados recursivos λ=0.90.

- **Caso 3:**

  Para λ=0.85

  Las gráficas pertenecientes a este caso están representadas a continuación, en la figura 54 se muestra la predicción y en la figura 55 los errores existentes.

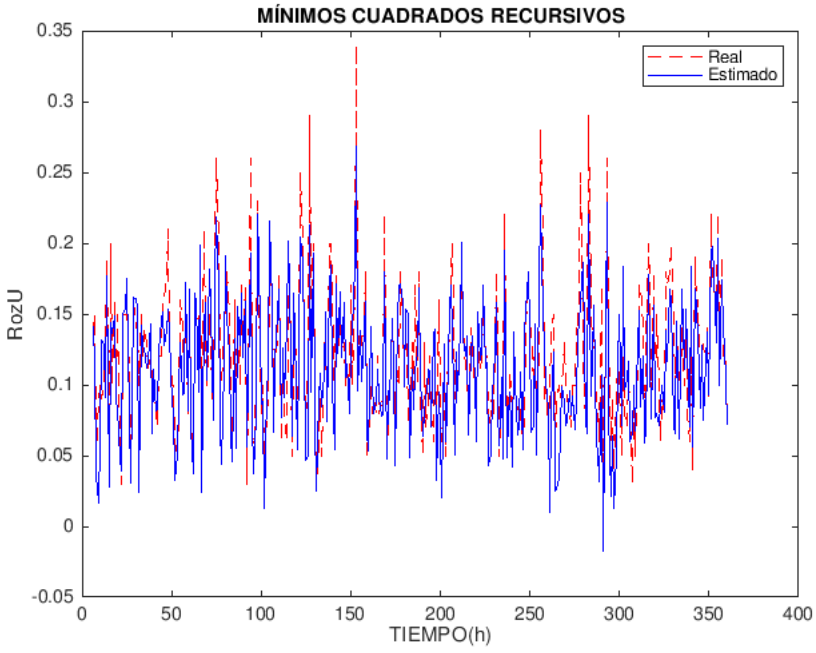


Figura: 54. Caso 3 mínimos cuadrados recursivos λ=0.85.

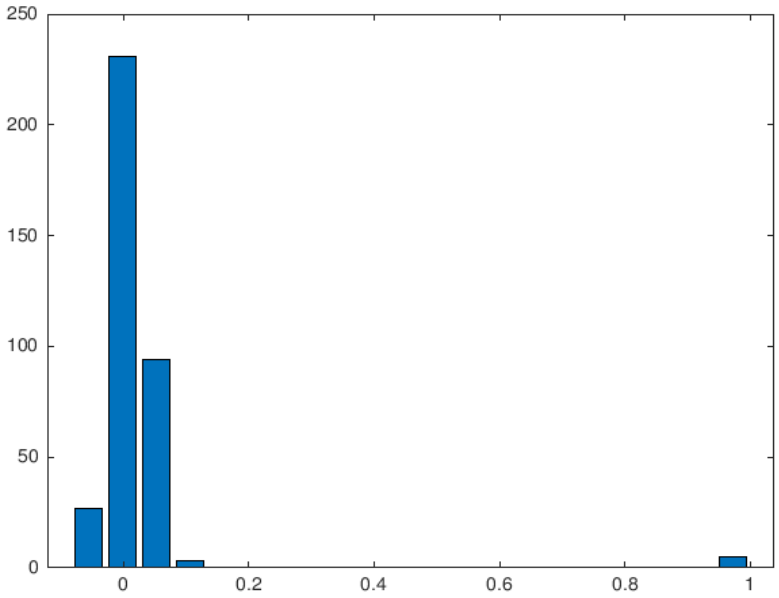


Figura: 55. Errores Caso 3 mínimos cuadrados recursivos λ=0.85.

	MAPE (%)	RMSE	MAE
Caso 3	14.26	0.0029	0.0211
Caso 3 (λ=0.95)	15.12	0.0030	0.0222
Caso 3 (λ=0.90)	14.31	0.0029	0.0225
Caso 3 (λ=0.85)	15.18	0.0029	0.0235

Tabla 6.Caso 3 Errores recursivos.

4.3.2.1 Conclusiones mínimos cuadrados recursivos

Se puede observar en las tablas creadas (tabla 4, 5,6) como en los tres casos empeora el MAPE (error porcentual absoluto medio), empeora considerablemente en todos los casos, mientras el resto de los errores se mantiene relativamente estable.

En los mínimos cuadrados recursivos, aparece el parámetro λ (factor de olvido), en el momento que su valor es la unidad provoca que el algoritmo tenga en cuenta para la estimación todos los valores de entrada y salida del sistema desde que inició su ejecución. Esto es perfectamente válido para la identificación de sistemas invariantes con el tiempo (cuyos parámetros no varían con el tiempo), pero se obtiene resultados erróneos si se usa con sistemas variantes. Para éstos conviene usar este factor de olvido que hace que el algoritmo tenga una memoria limitada

En el caso de este TFG, como vemos no tiene gran utilidad, esto es debido a que todavía no se poseen una gran cantidad de datos ni se ha dado anomalías como pudiera ser el

caso de una rotura por causas externas, un incendio o algún hecho por algún factor externo que de alguna manera falseara esos datos recogidos, en ese caso el sistemas se podría volver invariante con el tiempo, además estamos ante datos reales de robots que van progresando a nivel tecnológico a una gran velocidad por tanto en caso de realizar una actualización podría introducirse un factor de olvido hasta que el sistema se estabilizara durante un periodo de tiempo mayor.

Es por ello que en este caso los resultados son peores, y no tenga gran utilidad el uso de este factor de olvido, pero en la industria dada la gran capacidad de avance y el gran montante económico que las empresas automovilísticas destinan al I+D (investigación y desarrollo) hace que el sistema pueda variar su comportamiento en cuestión de semanas e incluso días.

Además, la introducción del factor de olvido provoca una mayor adaptación a las muestras es decir se produce una mayor sensibilidad y una mayor rapidez de adaptación, pero por el contrario es más insensible al ruido. En un momento puntual se puede conseguir que el algoritmo tenga muy alta sensibilidad a los valores actuales de los parámetros del sistema, aunque lamba sea igual a 1, y es reiniciando el valor de la matriz de covarianzas P a un valor semejante al que se le dio al comienzo del algoritmo. Al progresar en la identificación, los elementos de dicha matriz disminuyen su valor, provocando la insensibilidad del algoritmo a cambios en los parámetros.

Por tanto, en este TFG no se obtiene gran ventaja del uso de mínimos cuadrados recursivos con factor de olvido pero en gran volumen de datos y en cambios significativos en el sistema, podría llegar a tener gran utilidad su uso.

Capítulo 5

5 CÓDIGO

5.1 -ManiobrasJSON.java

```
package reader;
import java.io.File;
public class ManiobrasJSON {
    /*
     * arg[0] - Directory with all the the '.mdb' or '.accdb' files.
     * arg[1] - Connection string to connect with the MongoDB database.
     * arg[2] - Database name.
     * arg[3] - Collection name.
     */
    public static void main(String args[]) throws Exception {

        args = new String[]{"C:\\Users\\plazaro\\Desktop\\Maniobras", "pinzas-
maniobras", "doc"};

         File dir = new File(args[0]);

        if (dir.isDirectory()) {

         for (File f : dir.listFiles()) {

          new AccessReaderManiobrasJSON(f.getAbsolutePath()).generateJSONFile(args[1],
args[2]);
             }

         }

    }

}
```

5.2 -AccessReaderManiobrasJSON .java

```
package reader;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.healthmarketscience.jackcess.Column;
import com.healthmarketscience.jackcess.Database;
import com.healthmarketscience.jackcess.DatabaseBuilder;
import com.healthmarketscience.jackcess.Row;
import com.healthmarketscience.jackcess.Table;
import java.io.File;
import java.io.FileWriter;
import java.math.BigDecimal;
import java.util.Set;
import java.sql.Types;
import java.util.Date;
```

```
public final class AccessReaderManiobrasJSON {
    private String path;
    private File file;
    private Database database;

    public AccessReaderManiobrasJSON(String path) {
        setDatabase(path);
    }

    /*
     * ********************************
     * *** Getter and setter methods ***
     * ********************************
      public synchronized Database getDatabase() {
        return database;
    }

    /**
     * @brief Returns the Microsoft Access file absolute path.
     * @details Returns the Microsoft Access file absolute path.
     * @return The Microsoft Access file absolute path.
     */
    public synchronized String getPath() {
        return path;
    }

    /**
     * @brief Returns the Microsoft Access file object.
     * @details Returns the Microsoft Access file object (see the
'File' class).
     * @return The Microsoft Access file object.
     */
    public synchronized File getFile() {
        return file;
    }

    /**
     * @brief Opens a Microsoft Access file (the attached to the given
absolute
     * path) and push down the database content into a 'Database'
object.
     * @details Opens a Microsoft Access file (the attached to the
given
     * absolute path) and push down the database content into a
'Database'
     * object.
     * @param path The Microsoft Access file absolute path.
     */
    public synchronized void setDatabase(String path) {

        try {

            if (path == null) {
                throw new NullPointerException("The given path is
null.");
            } else if (!path.endsWith(".accdb") &&
!path.endsWith(".mdb")) {
                throw new Exception("The given path extension is not
'.accdb'.");
            } else {
```

```
                this.path = path;
            }

            file = new File(path);
            database = DatabaseBuilder.open(file);

            if (database == null) {
                throw new NullPointerException("The database reading
failed.");
            }

        } catch (Exception exception) {
            System.out.println("EXCEPTION >> " +
exception.getMessage());
        }

    }

    /*
     * *********************
     * *** Other methods ***
     * *********************
     */

    /**
     * @brief Returns the table with the specified name.
     * @details Returns the correspondent table according to the
specified name.
     * @return The specified table as a 'Table' instance.
     * @param tableName The given name of the table of interest.
     */
    public synchronized Table readTable(String tableName) {
        Table table = null;

        try {

            if (tableName == null) {
                throw new NullPointerException("The given table name
is null.");
            }

            table = database.getTable(tableName);
        } catch (Exception exception) {
            System.out.println("EXCEPTION >> " +
exception.getMessage());
        }

        return table;
    }

    /**
     * @brief Returns a tree display with a full view of the specified
table.
     * @details Returns a tree display with a full view of the
specified table.
     * @return A tree display with a full view of the specified table.
     * @param tableName The given name of the table to display.
     */
    public synchronized String getTableSummary(String tableName) {
        String summary = null;
```

```
        try {
            Table table = database.getTable(tableName);
            int i = 1, rows = table.getRowCount();

            if (table == null) {
                throw new NullPointerException("The given table is
null.");
            }

            summary = ">> Table " + table.getName() + " content\n";

            for (Row row : table) {
                System.out.println("Leyendo fila " + i + " de " + rows
+ ".\n");
                summary += "    >> Row " + i + "\n";

                for (Column column : table.getColumns()) {
                    summary += "       >> " + row.get(column.getName())
+ "\n";
                }

                i++;
            }

        } catch (Exception exception) {
            System.out.println("EXCEPTION >> " +
exception.getMessage());
        }

        return summary;
    }

    /**
     * @brief Returns a tree display with a full view of the complete
database.
     * @details Returns a tree display with a full view of the
complete
     * database.
     * @return A tree display with a full view of the complete
database.
     */
    public synchronized String getDatabaseSummary() {
        String summary = null;
        int i = 1;

        try {
            Set<String> tableNames = database.getTableNames();

            summary = "*** Database tree ***\n\n";

            for (String tableName : tableNames) {
                System.out.println("Leyendo tabla " + i + ".\n");
                summary += getTableSummary(tableName) + "\n";
                System.out.println("\n***\n\n");
                i++;
            }

        } catch (Exception exception) {
            System.out.println("EXCEPTION >> " +
exception.getMessage());
        }
```

```
        return summary;
    }

    /**
     * @brief Generates a JSON file based on the current M. Access
file data.
     * @details Generates a JSON file based on the current M. Access
file data.
     * If the name of it is 'MyFile.mdb', the resultant JSON file name
     * will be 'MyFile.json'. The structure of the JSON file will be
an array of
     * objects (one per row) without commas. Take care with the
floating point
     * numbers: they will be exported as strings (you can remove the
double
     * quotes later).
     * @param index The index of the objects.
     * @param type Deprecated in Kibana, but mandatory.
     */
    public synchronized void generateJSONFile(String index, String
type) {

        try {
            String halfName = path.substring(0, path.length() -
(path.endsWith(".accdb") ? 6 : 4));
            String jsonPath;
            String jsonText;
            String edited;
            Gson gson = new Gson();
            FileWriter fw;
            File jsonFile;
            Set<String> tableNames = database.getTableNames();
            Table table;
            JsonObject jo;
            Row row;
            Double d;
            Float f;
            Integer integer;
            Short s;
            BigDecimal bd;
            String tmps;
            Boolean b;
            Date date;

            for (String tableName: tableNames) {

                if (tableName == null) {
                    // throw new Exception("Null table name detected:"
+ tableName + ".");
                    continue;
                }

                table = database.getTable(tableName);

                if (table == null) {
                    // throw new Exception("Null table detected with
name:" + tableName + ".");
                    continue;
                }
```

```
                jsonPath = halfName + "_" + tableName;
                jsonText = "";

                for (int i = 0; i < table.getRowCount(); i++) {
                    System.out.println("Fila " + i + " de " +
table.getRowCount() + ".");
                    row = table.getNextRow();

                    if (row == null) {
                        // throw new Exception("Null row detected with
index: " + i + ".");
                        continue;
                    }

                    jo = new JsonObject();

                    for (Column column : table.getColumns()) {

                        if (column == null) {
                            // throw new Exception("Null column
detected.");
                            continue;
                        }

                        String cName = column.getName();

                        switch (column.getSQLType()) {
                            case Types.LONGNVARCHAR:
                                tmps = row.getString(cName);

                                if (tmps == null) {
                                    // throw new Exception("Null
string value.");
                                    continue;
                                }

                                jo.addProperty(cName, tmps);
                                break;

                            case Types.LONGVARCHAR:
                                tmps = row.getString(cName);

                                if (tmps == null) {
                                    // throw new Exception("Null
string value.");
                                    continue;
                                }

                                jo.addProperty(cName, tmps);
                                break;

                            case Types.VARCHAR:
                                tmps = row.getString(cName);

                                if (tmps == null) {
                                    // throw new Exception("Null
string value.");
                                    continue;
                                }

                                jo.addProperty(cName, tmps);
```

```
                    break;

                case Types.DATE:
                    date = row.getDate(cName);

                    if (date == null) {
                        // throw new Exception("Null
string value.");
                        continue;
                    }

                    tmps = date.toString();
                    jo.addProperty(cName, tmps);
                    break;

                case Types.TIMESTAMP:
                    date = row.getDate(cName);

                    if (date == null) {
                        // throw new Exception("Null
string value.");
                        continue;
                    }

                    tmps = date.toString();
                    jo.addProperty(cName, tmps);
                    break;

                case Types.INTEGER:
                    integer = row.getInt(cName);

                    if (integer == null) {
                        // throw new Exception("Null
integer value.");
                        continue;
                    }

                    jo.addProperty(cName, integer);
                    break;

                case Types.BIGINT:
                    bd = row.getBigDecimal(cName);

                    if (bd == null) {
                        // throw new Exception("Null big
decimal value.");
                        continue;
                    }

                    jo.addProperty(cName, bd);
                    break;

                case Types.SMALLINT:
                    s = row.getShort(cName);

                    if (s == null) {
                        // throw new Exception("Null short
value.");
                        continue;
                    }
```

```
                                jo.addProperty(cName, s);
                                break;

                            case Types.FLOAT:
                                f = row.getFloat(cName);

                                if (f == null) {
                                    // throw new Exception("Null float
value.");
                                    continue;
                                }


                if (cName.equals("ManPar") ||cName.equals("ManTot")){

jo.addProperty(cName,String.format("%.5f",f).replace(",", "."));
                                } else {
                                    jo.addProperty(cName, f);
                                }

                                break;

                            case Types.DOUBLE:
                                d = row.getDouble(cName);

                                if (d == null) {
                                    // throw new Exception("Null float
value.");
                                    continue;
                                }

                if (cName.equals("ManPar") || cName.equals("ManTot")){
    jo.addProperty(cName, String.format("%.5f", d).replace(",", "."));
                                } else {
                                    jo.addProperty(cName, d);
                                }

                                break;

                            case Types.DECIMAL:
                                d = row.getDouble(cName);

                                if (d == null) {
                                    // throw new Exception("Null float
value.");
                                    continue;
                                }


                if (cName.equals("ManPar") || cName.equals("ManTot")){

jo.addProperty(cName,String.format("%.5f", d).replace(",", "."));
                                } else {
                                    jo.addProperty(cName, d);
                                }

                                break;

                            case Types.BOOLEAN:
                                b = row.getBoolean(cName);
```

```
                                if (b == null) {
                                    // throw new Exception("Null float
value.");
                                    continue;
                                }

                                jo.addProperty(cName, b);
                            }

                        }

                        // Variable rozError.
                        JsonElement je = jo.get("RozU");
                        if (je != null) {
                jo.addProperty("RozError", Math.abs(je.getAsFloat()) > 0.5);
                        }

 /* Los campos NNewValue y NOldValue son números muy grandes,
 * tanto, que Gson los trata en notación científica. Para
 * que salgan todas las cifras hay que tomarlo como texto y,
 * posteriormente, editarlo del 'String' generado con los
 * objetos JSON. Por ello hay que localizar lo siguiente:
 * Sean:
 *     >> Anterior: la columna anterior a la de interés.
 *     >> X: la columna de interés (la del número grande).
 *     >> Posterior: la columna posterior a la de interés.
 *     >> valorAnt: valor para la columna 'Anterior'.
 *     >> valorPost: valor para la columna 'Posterior'.
*El objeto JSON contendrá una parte de la forma:
 *."Anterior":valorAnt,"X":"valorX","Posterior": vPost...
*Entonces, si queremos quitar las comillas del valor en
 * 'X', habría que hacer reemplazo del texto << "X":" >> por
 * << "X": >> y quitaríamos las comillas de apertura. Luego,
 * para las de clausura, habría que sustituir
 * << ","Posterior": >> por << ,"Posterior" >>.
                 */
            edited = gson.toJson(jo);
            edited = edited.replace("\"ManPar\":\"", "\"ManPar\":");
            edited = edited.replace("\",\"ManTot\"", ",\"ManTot\"");
            edited = edited.replace("\"ManTot\":\"", "\"ManTot\":");
            edited = edited.replace("\",\"ErrCod\"", ",\"ErrCod\"");
            edited = edited.replace("\\u003d", "=");
            edited = edited.replace("CET", "UTC");


                    jsonText +=
                        "{ \"index\" : { \"_index\" : \""
                        + index
                        + "-"
                        + tableName.toLowerCase()
                        + "\", \"_type\" : \""
                        + type
                        + "\" } }\n"
                        + edited
                        + "\n";


                if (i % 9999 == 0 || i == (table.getRowCount() - 1)) {
                        jsonFile = new File(jsonPath + i + ".json");
                        fw = new FileWriter(jsonFile);
```

```
                        /* File exporting. */
                        fw.write(jsonText);

                        /* File writer closure. */
                        fw.close();

                        jsonText = "";
                    }

                }

            }

        } catch (Exception exception) {
            System.out.println("EXCEPTION >> " +
exception.getMessage());
        }

    }

}
```

5.3 -Matlab: MINIMOS CUADRADOS

```
%la parte del código escrita en inglés, representa el trozo de código en
el cuál se entrenan los datos (conjunto de entrenamiento), por el
contrario, el código escrito en alemán representa la prueba de los datos
(conjunto de prueba)

Clc %limpieza ventana de comandos
clear all %limpieza del workspace
RozU=xlsread('RozU.xlsx'); %lectura del Excel con los valores del RozU
VarAbs=xlsread('VarAbs.xlsx'); %lectura Excel valores VarAbs
GrChError=xlsread('GrChError.xlsx'); %lectura Excel valores GrChError
RozU=RozU(:,1);
VarAbs=VarAbs(:,1);
GrChError=GrChError(:,1);
[m,n]=size(VarAbs);
Number_of_Days=7; %número de días de entrenamiento
day(1)=1; %mes en el cuál cogemos los días de entrenamiento
day(2)=25; %día en el cuál empieza el entrenamiento

%En el caso de la estimación por mínimos cuadrados de un periodo de la
pinza completo y con muestras por minuto, se utiliza el mismo código pero
cuando se multiplica días*horas hay que añadir *60 de cada minuto para
sacar una mayor cantidad de datos, el resto del código quedaría igual
ajustando las dimensiones de las matrices.

if day(1)==1
over=0;
end
if day(1)==2
over=24*20;
end
if day(1)==3
```

```
over=24*20+24*22;
end
locator=over+(day(2)-1)*24;
s=1;
for t=locator:(Number_of_Days*24+locator)
x0(s)=RozU(t-5*24);
x1(s)=RozU(t-24);
x2(s)=VarAbs(t);
x3(s)=GrChError(t);
sol(s)=RozU(t);
s=s+1;
end
x=[x0;x1;x2;x3;sol]';
intt=[ones(size(x0));x0;x1;x2;x3]';
[ndata,nvar]=size(x);
m=nvar-1; %nº de variables dependientes (x1,x2,…)
valuesx=x(1:ndata,1:m);
unos=ones(1,ndata);
R=[unos;valuesx']; %se introduce una fila de unos
C=R*R'; %se construye la matriz C
y=x(:,nvar); %la última columna es la variable independiente
b=R*y; %vector
[L,U,P]=lu(C); %Factorización de la matriz C, a partir de aquí se
resuelve el sistema lineal a=C*b
gamma=P*b;
ytr=inv(L)*gamma;
a=inv(U)*ytr;
Anzhal_der_Tages=7; %número de días de prueba
tage(1)=2; %mes en el cuál cogemos los días de prueba
tage(2)=14; %día en el cuál empieza el conjunto de prueba
if tage(1)==1
uber=0;
end
if tage(1)==2
uber=24*20;
end
if tage(1)==3
uber=24*20+24*22;
end
place=uber+(tage(2)-1)*24;
s=1;
for p=place:(Anzhal_der_Tages *24+place)
rp0(s)=RozU(p-5*24);
rp1(s)=RozU(p-24);
rp2(s)=VarAbs(p);
rp3(s)=GrChError(p);
rol(s)=RozU(p);
s=s+1;
end
r=[rp0;rp1;rp2;rp3;rol]';
intr=[ones(size(rp0));rp0;rp1;rp2;rp3]';
yt=rol';
[m,n]=size(yt);
MAPE=0; % se inicializa el error porcentual absoluto medio
```

```
RMSE=0; % se inicializa el root mean squared error
MAE=0; % se inicializa el %error absoluto medio
yest=intr*a;
errorp=ones(1,m);
for i=1:m
errorp(1,i)=(yr(i)-yest(i));
MAPE=MAPE+abs((yr(i)-yest(i))/yr(i)); %error porcentual absoluto medio
RMSE=sqrt((RMSE+(yr(i)-yest(i))^2)/m); %root mean squared error
MAE=MAE+(abs(yr(i)-yest(i))/m); %error absoluto medio
end
MAPE=MAPE/m;
errors=[MAPE RMSE MAE ]
time=1:m;
time=time';
plot(time,yr,'r--') %en color rojo aparecen los valores reales
hold on
plot(time,yest,'b') %en color azul aparecen los valores estimados
legend('Real','Estimado');
title('MÍNIMOS CUADRADOS');
ylabel('RozU');
xlabel('TIEMPO(h)');
figure %se genera una nueva figura para el diagrama de barras vertical
que muestra la cuenta de errores
[n,p] = hist(errorp,30);
bar(p,n);
```

5.4 -Matlab: MINIMOS CUADRADOS RECURSIVOS

```
Clc %limpieza ventana de comandos
clear all %limpieza del workspace
RozU=xlsread('RozU.xlsx'); %lectura del Excel con los valores del RozU
VarAbs=xlsread('VarAbs.xlsx'); %lectura Excel valores VarAbs
GrChError=xlsread('GrChError.xlsx'); %lectura Excel valores GrChError
RozU=RozU(:,1);
VarAbs=VarAbs(:,1);
GrChError=GrChError(:,1);
[m,n]=size(VarAbs);
Number_of_Days=7; %número de días de entrenamiento
day(1)=1; %mes en el cuál cogemos los días de entrenamiento
day(2)=25; %día en el cuál empieza el entrenamiento

%En el caso de la estimación por mínimos cuadrados de un periodo de la
pinza completo y con muestras por minuto, se utiliza el mismo código pero
cuando se multiplica días*horas hay que añadir *60 de cada minuto para
sacar una mayor cantidad de datos, el resto del código quedaría igual
ajustando las dimensiones de las matrices.

if day(1)==1
over=0;
end
if day(1)==2
over=24*20;
```

```
end
if day(1)==3
over=24*20+24*22;
end

for t=locator:(Number_of_Days*24+locator)
x0(s)=RozU(t-5*24);
x1(s)=RozU(t-24);
x2(s)=VarAbs(t);
x3(s)=GrChError(t);
sol(s)=RozU(t);
s=s+1;
end
x0=x0(:,1:s-2)';
x1=x1(:,1:s-2)';
x2=x2(:,1:s-2)';
x3=x3(:,1:s-2)';
sol=sol(:,1:s-2)';
% Calculo de la predicción mes siguiente
hours= Number_of_Days *24;% Método Persistente
RozU_est=nan(hours,1); % Método Mínimos Cuadrados
dimension_regression=4; % Dimensión del regresor
lambda=0.95 % Factor de olvido
Set_Theta=nan(hours,dimension_regression);
theta=(1/dimension_regression)*ones(dimension_regression,1);
P_covarianza_old=0.002*eye(dimension_regression);
for kk=1+dimension_regression:hours
m_act=[x0(kk) x1(kk) x2(kk) x3(kk)];
K=(P_old*m_act')/(lambda+m_act*P_old*m_act');
RozU_est (kk)=m_act*theta;
if (isfinite(RozU_est (kk))&&(isfinite(sol(kk))))
error_ls=sol(kk)- RozU_est (kk);
theta=theta+K*(error_ls);
P_covarianza_old=(1/lambda)*(eye(dimension_regression)-
K*m_act)*P_covarianza_old;
end
end
Anzhal_der_Tages =7;
tage(1)=2;
tage(2)=14;
if di(1)==1
uber=0;
end
if di(1)==2
uber=24*20;
end
if di(1)==3
uber=24*20+24*22;
end
end
place= uber+(tage(2)-1)*24;
s=1;
for p=place:(Anzhal_der_Tages *24+place)
rp0(s)=RozU(p-5*24);
```

```
rp1(s)=RozU(p-24);
rp2(s)=VarAbs(p);
rp3(s)=GrChError(p);
rol(s)=RozU(p);
s=s+1;
end
rol=rol(:,1:s-2)';
rp0=rp0(:,1:s-2)';
rp1=rp1(:,1:s-2)';
rp2=rp2(:,1:s-2)';
rp3=rp3(:,1:s-2)';
RozU_estt=[rp0 rp1 rp2 rp3]*theta;
hor=(2+dim_r):diax*24;
close all
plot(hor,rol((2+dimension_regression): Anzhal_der_Tages *24,:),'r--');
hold on;
plot(hor,RozU_estt((2+dimension_regression): Anzhal_der_Tages *24,:),'b')
legend('Real','Estimado');
title('MÍNIMOS CUADRADOS');
ylabel('RozU');
xlabel('TIEMPO(h)');
MAPE=0;
RMSE=0;
MAE=0;
errorp=ones(1, Anzhal_der_Tages *24);
for i=2+dimension_regression:horas
errorp(1,i)=rol(i)-RozU_estt(i);
MAPE=MAPE+abs((rol(i)-RozU_estt(i))/rol(i)); %error porcentual absoluto
medio
RMSE=sqrt((RMSE+(rol(i)-RozU_estt(i))^2)/( Anzhal_der_Tages *24)); %root
mean squared error
MAE=MAE+(abs(rol(i)-RozU_estt(i))/( Anzhal_der_Tages *24)); %error
absoluto medio
end
MAPE=MAPE/( Anzhal_der_Tages *24);
errors=[MAPE RMSE MAE]
figure
[n,p] = hist(errorp,30);
bar(p,n);
```

Capítulo 6

6 CONCLUSIONES

El principal objetivo de este TFG ha sido utilizar técnicas de aprendizaje automático para su implementación en el sector industrial en forma de mantenimiento predictivo. Estas técnicas requieren una gran cantidad de datos de datos históricos que permitan entrenar modelos, para este caso en concreto se tienen datos de los dos primeros meses de 2018. Fueron suficientes dado que los ciclos de vida de las pinzas robóticas no suelen superar los 15 días por ello se pudo crear modelos en diferentes espacios temporales y comprobar la confianza de los resultados obtenidos en otros diferentes.

Esto permite aprovechar el potencial de estas técnicas en el campo del mantenimiento predictivo, ya que aquello elementos con ciclos de vida más cortos, propensos a sufrir desgastes o disfunciones de uso son más adecuados para el uso de estas técnicas. Esto es debido a varios factores como se ha podido comprobar durante su realización:

- En su mayoría son más fácilmente medibles ya que se recopila una gran cantidad de información acerca del estado de gran número de variables del elemento. Al ser ciclos más cortos se dispone de más cantidad de información.
- Al ser elementos propensos a la rotura o el fallo transcurridos una serie de ciclos o número de operaciones, se requiere realizar mediciones periódicas para conocer su estado, además exige la necesidad de realizar mediciones en numerosos ciclos dado que, en caso de algún suceso anormal y aislado durante ese periodo, significaría no poder asumir como normal ese periodo de la pinza.

Una de las principales conclusiones del presente trabajo es la necesidad de rediseñar técnicas de recopilación y tratamiento de la información teniendo en cuenta en todo momento que información necesitamos procesar y en que formato para que pueda ser explotada en el futuro. Por ejemplo, en el caso de este TFG, se han utilizado hasta 3 formatos diferentes según la etapa en la que se encontrará y las necesidades que había que cubrir en ese momento. Por ello, en caso de conocer desde el principio las diferentes etapas y la metodología a trazar supone un ahorro importante de tiempo.

Con respecto al uso de los diferentes algoritmos, en el caso de uso realizado, basado en la regresión lineal, presenta un mayor grado de sencillez y agilidad en las predicciones, así como una gran flexibilidad en caso de tener que ajustar el modelo por alguna modificación en la instalación. Además, el reducido número de variables a analizar permitía realizar este tipo de estudio. En resumen, ante la necesidad de obtener una variable de salida continua, con un reducido número de variables interrelacionadas, unido a un porcentaje de error cometido bajo, aportando una buena precisión y

permitiendo observar claramente la tendencia del ciclo de vida de la pinza frente a su deterioro, inclinó la balanza hacia la regresión línea.

Como conclusión final, se puede afirmar que el uso de las técnicas de aprendizaje automático es viable, incluso puede realizarse de una manera más eficiente monitorizando las infraestructuras industriales de manera que la información sea recibida en tiempo real o conservando la información histórica para su posterior aprovechamiento.

Capítulo 7

7 MANUAL DE USUARIO

7.1 Guía de instalación

Se deben tener instalados una serie de elementos en el ordenador para poder ejecutar los programas creados:

- Sistema Operativo: Windows 10.
- Paquete Microsoft Office 365
- ElasticSearch (Kibana)
- JDK (javascript 8.0)
- Netbeans
- Comander (terminal, consola de comandos)
- Matlab

7.2 Manual de usuario

Los datos de las pinzas del robot se obtienen en Microsoft Access (.mdb), para poder abrirlos con el formato adecuado se ha de poseer una licencia Office (instalación gratuita en el caso del ámbito estudiantil), posteriormente para poder realizar un análisis en Kibana de estos datos, se procede a abrir el proyecto de Javascript en Netbeans (ambos descargables y de instalación gratuita).

Una vez introducido del código en Netbeans, se procede a ejecutar este código situándolo en el directorio adecuado, aquel en el que se encuentren los datos, con la finalidad de convertir el archivo .mdb en formato JSON, estructura de datos que requiere Kibana para su posterior análisis.

Antes de realizar el análisis en Kibana, se requiere subir los datos al logstash de ElasticSearch, para ello se ha progamado en bash un curl que ejecutando en el comander (consola de comandos), procede a subir a la nube todos los datos, situando en el directorio adecuado para su correcta subida.

Una vez analizados los datos en Kibana se exportan los datos a formato Excel (incluido en el paquete Office) dado que Matlab (su instalación es gratuita en versión de estudiante, y en el caso de este TFG se ha utilizado el 2018 online con las toolbox que éste agrega). Es necesario el formato Excel dado que Matlab lee este formato, en este

caso se lee el archivo, y posteriormente se exporta aquella columna que interesa de cada fichero.

Por último, una vez instalado Matlab sólo falta por introducir el código que aparece anexado, tanto el de mínimos cuadrados como el de mínimos cuadrados recursivos. En todo ese código, los únicos parámetros que el usuario debe ajustar son aquellos que determinan el periodo de entrenamiento y el periodo de prueba, determinando el día de comienzo, el día de fin, y su duración.

Capítulo 8

8 PRESUPUESTO

A continuación, se adjunta un desglose de los costos estimados para este proyecto. El presupuesto se ha separado en costes hardware, costes software y costes por mano de obra.

8.1 Costes hardware

Concepto	Precio	Unidades	Total
Portatil Intel Core i7-3610QM	600 €	1	600 €
Monitor 24''	115€	1	115 €
Disco Duro Externo 2Tb	85€	1	85 €
Ratón inalámbrico	20€	1	20 €
Total Hardware			**820€**

Tabla 7. Costes hardware.

8.2 Costes software

Concepto	Precio	Unidades	Total
Windows 10	100€	1	100€
Matlab Online R2018a	0€	1	0€
Microsoft Office 365	0€	1	0€
JavaScript 8.1	0€	1	0€
Netbeans	0€	1	0€
ElasticSearch(Kibana)	1000€	1	1000€
MongoDB	0€	1	0€
Total Software			**1100€**

Tabla 8.Costes software.

En el desglose del presupuesto la gran parte del software aparece como gratuito, en gran medida se debe o al hecho del que software es gratuito o como el caso de Matlab y el paquete Office ofrecen una versión gratuita a estudiantes, y esa versión fue la escogida de cara a realizar el proyecto de una manera más económica.

8.3 Costes mano de obra

Concepto	Precio/hora	Nº horas	Total
Ingeniería	30€	640	19.200€
Instalación	24€	15	360 €
Testing	28€	200	5.600 €
Documentación	25€	140	3.500 €
Total mano de obra			**28.660€**

Tabla 9.Costes mano de obra.

A pesar de que todo el trabajo ha sido realizado por la misma persona, el trabajo ha sido dividido en cuatro perfiles de trabajo diferentes. Los conceptos de "Ingeniería" son el tiempo invertido en investigación, aplicación de los conocimientos de robótica y física para poder entender el funcionamiento de los robots y las pinzas en contacto con la chapa o la parte de la carrocería del automóvil, "Instalación" se refiere al proceso de instalación del software en el equipo usado durante el TFG, "Prueba" a las pruebas finales que son necesarias comprobar que el sistema funciona correctamente y el formato conseguido en lenguaje java compila correctamente, y que el algoritmo programado en Matlab se ajusta a las especificaciones requeridas y el error admitido, y por último "Documentación" el tiempo invertido en el desarrollo y la ordenación del documento escrito.

8.4 Costes totales

Concepto	Total
Costes Hardware	820€
Costes Software	1.100€
Costes mano de obra	28.660€
Total mano de obra	**30.580€**

Tabla 10.Costes totales.

8.5 Presupuesto contractual

El presupuesto del contrato es el costo total más un 13% adicional (costos generales) y un 6% (beneficios industriales).

8.6 Monto total del presupuesto

Finalmente, para el presupuesto del contrato que se ha calculado antes, se debe agregar el porcentaje de IVA (impuesto de valor añadido), que en España tiene un porcentaje del 21%.

Concepto	Total
Costes totales	30.580€
Costes generales	3.975,40€
Beneficios industriales	1.834,80€
Presupuesto contractual	**36.390,2€**

Tabla 11.Monto total.

Concepto	Total
Presupuesto contractual	36.390,2€
Costes IVA (21%)	7.641,94€
Total mano de obra	**44.032,14€**

Tabla 12.Monto total con IVA.

El costo final del proyecto es CUARENTA Y CUATRO MIL TREINTA Y DOS euros y CATORCE céntimos, 44032,14€.

Madrid, 23 June 2018
Firmado: Pablo Lázaro Enguita

Capítulo 9

9 BIBLIOGRAFÍA Y REFERENCIAS

[1] Brett Lantz, "**Machine Learning with R**", *PACKT, 2012.*

[2] Shai Ben-David, Shai Shalev-Shwartz, "**Understanding Machine Learning: From Theory to Algorithms**", *LEARNING*, *2013.*

[3] Coursera Course Standford University "**https://es.coursera.org/learn/machine-learning**" *(fecha de consulta: 15 de febrero de 2018).*

[5] Michael Paluszek, Stephanie Thomas "**MATLAB Machine Learning**", *APRESS, 2016.*

[6] Astor de Caso Parra, "**JavaScript**", *ANAYA, EDICION 2018.*

[7] Machtelt Garrels, "**Bash Guidesfor Beginners**", *FULTURS, 2004.*

[8] Ethem Alpaydin, "**Introduction to Machine Learning**", *STRUCTURE, 2004.*

[9] Kevin P.Murphy, "**Machine Learning. A propabilistic Perspective**", *PACKT, 2012.*

[10] Julia Garriga Trillo, José María Merino, Juan Carlos Suárez Falcón, Miguel Padilla Suárez, Patricia Recio Saboya y Paula Lubin Pigouche, "**Introducción al análisis de datos**", *UNED, 2009.*

[11] Manuel Gil Rodriguez, "**Introducción rápida a MatLab y Simulink para ciencia e ingeniería**", *DIAZ DE SANTOS, 2003.*

[12] Amos Gilat, "**Matlab: una introducción con ejemplos prácticos**", *2000.*

[12] Clinton Gormley, Zachary Tong, "**Elasticsearch: The Definitive Guide: A Distributed Real-Time Search**", *O' REILLY, 2015.*

[13] Bahaaldine Azarmi , "**LEARNING KIBANA 5.0**", *PACKT, 2017.*

[14] Yuvraj Gupta, "**Kibana Essentials**", *PACKT, 2015.*

[15] Alasdair Gilchrist, "**Industry 4.0: The Industrial Internet of Things**", *APRESS, 2016.*

[16] Rajesh Agnihotri, Samuel New, "**Industry 4.0: DATA ANALYTYCS**", *2005*

[17] Stephen Marsland, "**Machine Learning: An Algorithmic Perspective,**", *CRC PRESS, 2009.*

[18] Alp Ustundag, Emre Cevikcan, "**Industry 4.0: Managing the digital transformation**", *SRPINGER, 2008.*

[19] Byron Ellis, "**Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data**", *WILEY, 2014.*

[20] Sumit Gupta, "**Real-Time Big Data Analytics**", *PACKT, 2016.*

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá